2017

Knowledge is power: Quantum chemistry on novel computer architectures

Kristopher William Keipert
Iowa State University

Knowledge is power: Quantum chemistry on novel computer architectures

by

Kristopher William Keipert

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Physical Chemistry

Program of Study Committee:
Mark S. Gordon, Major Professor
Theresa L. Windus
Monica H. Lamm
Jacob W. Petrich
Xueyu Song

Iowa State University

Ames, Iowa

2017

DEDICATION

This work is dedicated to my daughter Madeleine.

TABLE OF CONTENTS

ABSTRACT

In the first chapter of this thesis, a background of fundamental quantum chemistry concepts is provided. Chapter two contains an analysis of the performance and energy efficiency of various modern computer processor architectures while performing computational chemistry calculations. In chapter three, the processor architectural study is expanded to include parallel computational chemistry algorithms executed across multiple-node computer clusters. Chapter four describes a novel computational implementation of the fundamental Hartree-Fock method which significantly reduces computer memory requirements. In chapter five, a case study of quantum chemistry two-electron integral code interoperability is described. The final chapters of this work discuss applications of quantum chemistry. In chapter six, an investigation of the esterification of acetic acid on acid-functionalized silica is presented. In chapter seven, the application of ab initio molecular dynamics to study the photoisomerization and photocyclization of stilbene is discussed. Final concluding remarks are noted in chapter eight.

CHAPTER 1: INTRODUCTION

General Overview

The field of theoretical quantum chemistry is a pursuit to describe chemical properties by application of quantum mechanical principles. Applications of theoretical quantum chemistry have complemented experimental studies with fundamental explanations of observed physical phenomena and prediction of physical properties which are experimentally unobtainable. In practice, the modeling of chemical systems is achieved by using complex computer programs to compute approximate solutions to the Schrödinger equation. Computational chemists develop the expertise to choose the appropriate approximations and theoretical models required to accurately compute the chemical properties of interest. As the performance of computer hardware architecture progresses, computational chemistry programs are used to model increasingly large chemical systems at unprecedented levels of accuracy.

Dissertation Organization

In the first chapter of this thesis, a background of fundamental quantum chemistry concepts is provided. Chapter two contains an analysis of the performance and energy efficiency of various modern computer processor architectures while performing computational chemistry calculations. In chapter three, the processor architectural study is expanded to include parallel computational chemistry algorithms executed across multiple-node computer clusters. Chapter four describes a novel computational implementation of the fundamental Hartree-Fock method

which significantly reduces computer memory requirements. In chapter five, a case study of quantum chemistry two-electron integral code interoperability is described. The final chapters of this work discuss applications of quantum chemistry. In chapter six, an investigation of the esterification of acetic acid on acid-functionalized silica is presented. In chapter seven, the application of *ab initio* molecular dynamics to study the photoisomerization and photocyclization of stilbene is discussed. Final concluding remarks are noted in chapter eight.

Theoretical Background

The evolution of quantum systems over time is described by the time-dependent Schrödinger equation[1]:

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \hat{H}\Psi(\mathbf{r},t) \tag{1}$$

for which i is the imaginary unit, $\hbar$ is the reduced Planck constant, Ψ is the wavefunction of the quantum system with position vector $\mathbf{r}$ (e.g. nuclear and electronic coordinates) at time t, and $\hat{H}$ is the Hamiltonian operator[1]. Much of quantum chemistry addresses atomic and molecular orbital stationary state solutions to the Schrödinger equation. If a Hamiltonian without explicit time dependence is employed, then the general time-independent Schrödinger equation can be derived and expressed as the eigenvalue equation

$$\hat{H}\Psi(\mathbf{r}) = E\Psi(\mathbf{r}) \tag{2}$$

for which E is the total energy of the quantum system. For a system comprised of N electrons and M nuclei, the molecular Hamiltonian operator is defined as

[1] The units used throughout this work are atomic units, unless noted otherwise.

$$\hat{H} = -\sum_{i=1}^{N}\frac{\nabla_i^2}{2} - \sum_{A=1}^{M}\frac{\nabla_A^2}{2M_A} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|r_i - R_A|} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{|r_i - r_j|} + \sum_{A=1}^{M}\sum_{B>A}^{M}\frac{Z_A Z_B}{|R_A - R_B|} \quad (3)$$

for which M_A is the mass of nucleus A divided by the mass of an electron, Z_A is the nuclear charge of nucleus A, $\mathbf{r}_i$ and $\mathbf{r}_j$ are the coordinates of electrons *i* and *j*, $\mathbf{R}_A$ and $\mathbf{R}_B$ are the coordinates of nuclei A and B, and ∇^2 is the Laplacian operator with respect to particle coordinates. In order, the five terms of the molecular Hamiltonian represent the electronic kinetic energy ($\hat{T}_e$), the nuclear kinetic energy ($\hat{T}_N$), the electron-nucleus attraction potential energy ($\hat{V}_{eN}$), the electron-electron repulsion potential energy ($\hat{V}_{ee}$), and the nuclear-nuclear repulsion potential energy ($\hat{V}_{NN}$). The coupling of electronic and nuclear motions in the $\hat{V}_{eN}$ term prevents separation of nuclear and electronic coordinates, and so computation of the wavefunction is an intractable many-body problem beyond trivial cases. In practice, these motions are usually assumed to be separable by invoking the Born-Oppenheimer approximation[2]. Since electrons are much lighter than nuclei, electronic motion can be approximated as instantaneous response to fixed nuclear positions. Under this assumption, the value of $\hat{T}_N$ is negligible and the $\hat{V}_{NN}$ term shifts energy eigenvalues by a constant factor. These terms are neglected from Eq. 3 to define an electronic Hamiltonian operator, $\hat{H}_{el}$

$$\hat{H}_{el} = -\sum_{i=1}^{N}\frac{\nabla_i^2}{2} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|r_i - R_A|} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{|r_i - r_j|} \quad (4)$$

which depends only parametrically on nuclear coordinates. The electronic Hamiltonian is used to compute an electronic wavefunction Ψ_{el} by

$$\hat{H}_{el}\Psi_{el}(\mathbf{r}) = E_{el}\Psi_{el}(\mathbf{r}) \quad (5)$$

, and the electronic energy obtained is used as a potential for a nuclear wavefunction. As such, nuclear motions are defined by the nuclear kinetic energy, the nuclear-nuclear electrostatic repulsions and the fixed-nucleus electronic energy.

Solving the electronic Schrödinger equation is a prerequisite for computing a potential energy surface. The electronic wavefunction provides the electron probability density via the Born postulate and consequently various molecular properties as well. Unfortunately, examination of the electronic Hamiltonian reveals an underlying many-body problem in the pairwise electronic interaction of the electron-electron repulsion term. The electron-electron repulsion term is inseparable due to the inter-electron distance, so an analytical solution for the electronic Schrödinger equation is not possible for systems with multiple electrons. Neglecting the term is a poor approximation, because electron repulsion contributes significantly to the total electronic energy and correlates the spatial occupation of electrons. In Hartree-Fock theory[3], $\hat{V}_{ee}$ is replaced with a one-electron potential that captures the repulsion between an electron and the mean field of all other electrons (v^{HF}). Now, the exact electronic Hamiltonian $\hat{H}_{el}$ can be approximated as a sum of one-electron operators. Before v^{HF} is explicitly defined, it is useful to describe the form of a multielectron wavefunction composed of independent one-electron wavefunctions.

A spin orbital is a wavefunction $\chi(\mathbf{x})$ that describes the spatial and spin components of one electron. The Hartree Product Ψ^{HP} is a simple N-particle wavefunction constructed from the product of N independent spin orbitals.

$$\Psi^{HP}(\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N) = \chi_1(\mathbf{x}_1)\chi_2(\mathbf{x}_2)\mathrm{L}\ \ \chi_N(\mathbf{x}_N) \tag{6}$$

The Hartree product wavefunction is symmetric with respect to exchange of spatial and spin coordinates of two electrons, so it is not a valid wavefunction for fermionic systems. The Pauli exclusion principle can be satisfied by taking a linear combination of Hartree products. For a two electron system, the proper antisymmetric linear combination is

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}(\chi_1(\mathbf{x}_1)\chi_2(\mathbf{x}_2) - \chi_1(\mathbf{x}_2)\chi_2(\mathbf{x}_1)) \tag{7}$$

which can be written as a Slater determinant[4]

$$\Psi(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{\sqrt{2}}\begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_2(\mathbf{x}_1) \\ \chi_1(\mathbf{x}_2) & \chi_2(\mathbf{x}_2) \end{vmatrix} \tag{8}$$

and generalized for an N-electron system.

$$\Psi(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \chi_1(\mathbf{x}_1) & \chi_2(\mathbf{x}_1) & \cdots & \chi_N(\mathbf{x}_1) \\ \chi_1(\mathbf{x}_2) & \chi_2(\mathbf{x}_2) & \cdots & \chi_N(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \chi_1(\mathbf{x}_N) & \chi_2(\mathbf{x}_N) & \cdots & \chi_N(\mathbf{x}_N) \end{vmatrix} \tag{9}$$

The wavefunction in (9) is used as the Hartree-Fock wavefunction, Ψ^{HF}. By definition, the ground state energy obtained from an approximate wavefuction is higher than the exact energy. The Hartree-Fock method is used to variationally solve for an optimal set of spin orbitals which minimizes the energy. It can be shown that the optimal set of orbitals each satisfy the eigenvalue equation

$$\hat{f}\chi(\mathbf{x}_i) = \epsilon_i\chi(\mathbf{x}_i) \tag{10}$$

where $\hat{f}$ is the one-electron Fock operator, and the eigenvalue ϵ_i is the orbital energy.

The Fock operator consists of the one-electron terms of the Hamiltonian, and the previously mentioned Hartree-Fock potential.

$$\hat{f} = -\frac{\nabla_i^2}{2} - \sum_{i=1}^{N}\sum_{A=1}^{M}\frac{Z_A}{|\mathbf{r}_i - \mathbf{R}_A|} + v^{HF}(i) \tag{11}$$

The potential for closed shells is now explicitly defined as

$$v^{HF}(i) = \sum_{j=1}^{N/2}(2\hat{J}_j - \hat{K}_j) \tag{12}$$

where $\hat{J}$ is the coulomb operator

$$\hat{J}_j\chi_i(\mathbf{x}_1) = \left[\int d\mathbf{x}_2 \chi_j^*(\mathbf{x}_2)\chi_j(\mathbf{x}_2)r_{12}^{-1}\right]\chi_i(\mathbf{x}_1) \tag{13}$$

and $\hat{K}$ is the exchange operator.

$$\hat{K}_j\chi_i(\mathbf{x}_1) = \left[\int d\mathbf{x}_2 \chi_j^*(\mathbf{x}_2)\chi_j(\mathbf{x}_1)r_{12}^{-1}\right]\chi_i(\mathbf{x}_2) \tag{14}$$

Because of the dependence on the molecular orbitals in the Fock operator, (10) must be solved iteratively. The spatial component of a molecular orbital is usually approximated by a linear combination of basis functions called atomic orbitals ϕ_μ,

$$\psi_i = \sum_{\mu=1} C_{\mu i}\phi_\mu \tag{15}$$

where C is a contraction coefficient. Gaussian basis functions[5] are commonly used; they may consist of a single Gaussian function or multiple contracted Gaussian functions. In the basis set representation, the integro-differential Hartree-Fock equations are represented in matrix form by the Roothaan-Hall equations[6]

$$\mathbf{FC} = \mathbf{SC}\epsilon \tag{16}$$

where **F** is the Fock matrix defined by the Fock operator, **C** is a matrix of molecular orbital coefficients, **S** is the overlap matrix of the basis functions, and ϵ is a diagonal matrix containing

the orbital energies. The algebraic Roothaan-Hall equations can be solved by standard matrix techniques.

The Hartree-Fock method typically accounts for more than 99% of the exact relativistic energy E_0. The unrecovered energy is called the correlation energy, E_{corr}.

$$E_{corr} = E_0 - E_{HF} \tag{17}$$

Although the magnitude of the correlation energy is relatively small, the inclusion of correlation effects is crucial for accurate description of chemical bonding and electronic excitation, among other important phenomena[2]. The correlation energy can be divided into two distinct categories. Dynamic correlation energy arises from correlated electron motion via instantaneous Columbic repulsion. Non-dynamic or static correlation energy reflects the inadequacy of the single Slater determinant in accurately describing molecular systems with multiple nearly degenerate electronic configurations. As both types of correlation energy come from the same physical phenomenon, the distinction is artificial. A number of "post-Hartree-Fock" or "correlated" methods have been developed in order to improve Hartree-Fock by recovering correlation energy. One commonly used approach is Møller–Plesset perturbation theory[7], in which a perturbation V is added to the unperturbed Hartree-Fock Hamiltonian $\hat{H}_0$:

$$\hat{H}_{el} = \hat{H}_0 + \lambda V \tag{18}$$

The exact eigenfunctions and eigenvalues are expanded in a Taylor series in λ

$$\Psi^{(n)} = \Psi^{(0)} + \sum_{i=1}^{n} \lambda^i \Psi^{(i)} \tag{19}$$

[2] Correlation energy in this text specifically refers to Coulomb correlation. The Hartree-Fock electron exchange term fully accounts for Fermi correlation.

$$E^{(n)} = E^{(0)} + \sum_{i=1}^{n} \lambda^i E^{(i)} \tag{20}$$

The perturbation is truncated at the chosen n^{th} order (PTn). The series often does not converge, so the perturbation is commonly truncated at second order. Including higher order terms significantly increases computational cost with an unpredictable impact on accuracy. The MP2 energy expression is

$$E_{MP2} = \frac{1}{4} \sum_{i,j,a,b} \frac{\left\langle \phi_i \phi_j \,|\, V \,|\, \phi_a \phi_b \right\rangle \left\langle \phi_a \phi_b \,|\, V \,|\, \phi_i \phi_j \right\rangle}{\epsilon_i + \epsilon_j - \epsilon_a - \epsilon_b} \tag{21}$$

The *i* and *j* indices run over occupied orbitals, and the *a* and *b* indices run over virtual orbitals. MP theory is size consistent but not variational - the computed energy may be lower than the actual ground state energy. MP2 typically recovers 80-90% of the correlation energy, with a computational cost scaling at order n^5 relative to system size. Perturbation theory is most successful when a small perturbation is applied, which means the Hartree-Fock wavefunction should be a good approximation of the exact wavefunction. As such, the reliability of MP methods is very system dependent.

In other popular correlated methods, a multi-determinant electronic wavefunction is systematically constructed by combining multiple configuration state functions (CSFs) built from spin orbitals. The resulting wavefunction includes excited determinants relative to a reference such as the Hartree-Fock wavefunction. One approach is the configuration interaction method (CI)[8]. The CI wavefunction Ψ^{CI} is expanded by applying a linear excitation operator C to the reference wavefunction.

$$\Psi^{CI} = (1 + C)\Psi^{HF} \tag{22}$$

$$C = \sum_{j=1}^{N} C_j \tag{23}$$

The excitation operator subscript refers to the number of excited electrons. For example, $C_2 \mid \phi_{HF} \rangle$ represents all terms involving double excitations. The weight (CI coefficient) for each determinant is variationally optimized. If the full CI expansion is used, then the exact non-relativistic energy can be computed within the Born-Oppenheimer and basis set limitations. Full CI is currently impractical for more than ~10 electrons with a moderate basis set, as the computational cost of the method scales factorially with respect to system size. In practice, the CI expansion is typically truncated to include only double excitations (CID), or single and double excitations (CISD). Energy eigenvalues for ground and excited electronic states are computed during the CI procedure, so the method can be used to calculate excitation energies.

While full CI is size extensive and size consistent, the properties are lost when the expansion is truncated. A class of methods based on the related but distinct coupled-cluster (CC) theory[9] is size extensive and size consistent even after truncation. The main difference between CC and CI is the form of the excitation operator. In CC theory, an exponential cluster operator e^T acts on the reference wavefunction

$$\begin{aligned} \Psi^{CC} &= e^{(T_1+T_2+T_3\ldots)}\Psi^{HF} \\ &= (1 + T_1 + T_2 + T_3 + \frac{1}{2}T_1^2 + T_2T_1 + \frac{1}{2}T_3^2 + \frac{1}{6}T_1^3 + \ldots)\Psi^{HF} \end{aligned} \tag{24}$$

where T_n is the operator of all *n*-electron excitations. The cluster operator is written in terms of creation operators $\hat{a}^a$ and annihilation operators $\hat{a}_i$ acting on occupied orbitals (hole states) *i,j* and virtual orbitals (particle states) *a,b*, and excitation amplitude coefficients *t*.

$$T_n = \frac{1}{(n!)^2} \sum_{i_1,i_2,\ldots,i_n} \sum_{a_1,a_2,\ldots,a_n} t_{a_1,a_2,\ldots,a_n}^{i_1,i_2,\ldots,i_n} \hat{a}^{a_1}\hat{a}^{a_2}\ldots\hat{a}^{a_n}\hat{a}_{i_n}\ldots\hat{a}_{i_2}\hat{a}_{i_1} \qquad (25)$$

CC methods are computationally demanding, and inclusion of single, double, and triple excitations is usually required for accurate energy calculations. With respect to system size, CCSD scales as N^6 and CCSDT scales as N^8. One approach to reducing the computational cost is combining the standard iterative singles and doubles method (CCSD) with an estimation of the T_3 energy by perturbation theory (CCSD(T))[10], which reduces the scaling compared to full CCSDT by an order of magnitude. CC and CI theory are equivalent if the excitation operators are not truncated, but when truncated at the same excitation level, CC methods include contributions from higher-order terms which are missing from the analogous CI expansion. The additional contributions are products of lower-order terms. For example, the expansion of the T_1 and T_2 operators is:

$$e^T = 1 + \{T_1\} + \left\{T_2 + \frac{1}{2}T_1^2\right\} + T_1T_2 + \frac{1}{2}T_2^2 \qquad (26)$$

The first and second bracketed terms are analogous to the C_1 and C_2 terms of the CI excitation operator expansion, respectively. While the computational cost of CC and CI scale similarly at the same operator truncation level, these extra terms partially contribute to the larger scaling prefactor of CC methods, and to the size consistence of the CC method.

The truncated CI and CC methods are among the most practical and highly accurate tools used by computational chemists over the past few decades. While many molecular systems can be successfully studied with these methods, they are based on a single-reference determinant and often fail to correctly describe systems with significant static correlation (e.g. homolytic bond

cleavage, nearly-degenerate electronic states). Even for well-separated states, truncated CI tends to overestimate excitation energies because the ground and excited states are not equivalently correlated[11]. Multi-determinant reference wavefunctions are commonly computed with the multi-configurational self-consistent field (MCSCF)[12] method. The MCSCF wavefunction is a truncated CI expansion in which the molecular orbitals for each configuration are optimized in addition to the CI coefficients. The choice of which configurations to include is an important consideration and highly dependent on the molecular system and physical properties of interest. One approach is complete active space SCF (CASSCF)[13], in which orbitals are partitioned into core, active, and virtual spaces. Core and virtual orbitals are the same as Hartree-Fock occupied and virtual orbitals. Active orbitals within a chosen active space can be partially occupied. Within the active space, a full CI calculation is performed which generates determinants for every possible electron configuration within the space. Considering the computational cost of full CI, it is useful to choose the smallest active space that can sufficiently model the problem at hand. Once the MCSCF wavefunction is obtained, it may be used as a reference for other methods to compute dynamic correlation energy. Common choices include multi-reference CI[14] or multi-reference perturbation methods such as complete active space perturbation theory[15]. In addition to "true" multi-reference methods, some single-reference approaches have been developed to describe limited static correlation effects[16].

As a crude summary, the Hartree-Fock method is based on the idea that a many-electron wavefunction can be represented by a determinant of one-electron wavefunctions. The error introduced by this assumption within the nonrelativistic Born-Oppenheimer regime, the correlation energy, is physically important but computationally expensive to recover. This

expense comes from the requirement to emulate the correlated electron-electron motion neglected in Hartree-Fock theory. In 1964, Kohn and Hohenberg introduced density functional theory (DFT)[17], which is today the most widely used computational chemistry method. Many functionals of the electron density have been developed which allow computation of various molecular properties. While a wavefunction is a 3N-dimensional variational problem, the electronic density depends on only 3 spatial coordinates for any number of electrons. As a result, the computational scaling of DFT is similar to Hartree-Fock with the benefit of recovering correlation energy.

Kohn-Sham DFT (KS-DFT)[18] is the most commonly used implementation of DFT. The Kohm-Sham equations are a coupled set of differential equations which define a one-particle Schrödinger-like equations similar to the Hartree-Fock equations. As in Hartree-Fock theory, the Kohm-Sham equations describe a system of independent particles subject to an external potential. The effective one-electron Hamiltonian operator used in KS-DFT is the Kohm-Sham operator, $\hat{h}_{KS}$.

$$\hat{h}_{KS} = -\frac{\nabla^2}{2} + \sum_{A=1}^{M} \frac{Z_A}{|r - r_A|} + \int \frac{\rho(r')}{|r - r'|} dr' + V_{xc} \tag{27}$$

The exchange-correlation potential $V_{xc}(r)$ and the corresponding energy expression are the only unknown variables in KS-DFT. If the exact $V_{xc}(r)$ were known, then the energies computed by KS-DFT would be exact. The lack of exact exchange and correlation functionals is the major shortcoming of DFT. Many approximate functionals have been developed, often by empirically fitting to data obtained from experiment or wavefunction *ab initio* methods. As a result, the

accuracy of properties computed with DFT is dependent on the chosen exchange-correlation functional. When a functional is carefully chosen for the problem of interest, DFT can be a powerful tool with relatively low computational cost.

In 1937, physicist Howard Hathaway Aiken wrote:

"At the present time, there exist problems beyond our ability to solve, not because of theoretical difficulties, but because of insufficient means of mechanical computation."[19]

While the quantum chemistry methods discussed so far are theoretically robust, there are two primary limitations which restrict the scope of tractable problems. First, there is a limited number of floating point operations per second (FLOPS) that current computer hardware can perform. Since 2008, the fastest supercomputers running at peak performance are capable of throughput at the magnitude of petaflops[20]. Reaching exascale performance will require significant advances in energy efficiency, memory, and interconnect technology. The second limitation lies in the efficient implementation of quantum chemistry methods on supercomputers. Computer hardware and software tool technologies are moving targets which advance rapidly compared to computational chemistry software. Achieving strong application performance on modern supercomputers often requires careful restructuring of core computational kernels.

References

1. (a) Schrödinger, E. *Ann. Phys.* **1926**, *79*, 361. (b) Schrödinger, E. *Ann. Phys.* **1926,** *79*, 489. (c) Schrödinger, E. *Ann. Phys.* **1926**, *79*, 734. (d) Schrödinger, E. *Ann. Phys*. **1926**, *80*, 437. (e) Schrödinger, E. *Ann. Phys.* **1926**, *81*, 109. (f) Schrödinger, E. *Naturwissenschaften* **1926**, *14*, 664.
2. Born, M.; Oppenheimer, R. *Ann. Phys.* **1927**, *389*, 457.
3. (a) Hartree, D.R. *Math. Proc. Cambridge Philos. Soc.* **1928**, *24*, 89 (b) Fock, V. *Z. Phys.* **1930**, *61*, 126. (c) Fock, V. *Z. Phys.* **1930**, *62*, 795.
4. Slater, J.C. *Phys. Rev.* **1929**, *34*, 1293.
5. Boys, S.F. *Proc. R. Soc. London Ser. A*. **1950**, *200*, 542.
6. (a) Roothan, C.C.J. *Rev. Mod. Phys.* **1951**, *23*, 69. (b) Hall, G.G. *Proc. R. Soc. A* **1951**, *205*, 541.
7. Møller, C.; Plesset, M.S. *Phys. Rev*. **1934**, *46*, 618.
8. (a) Kellner, G.W. *Z. Phys.* **1927**, *44*, 91. (b) Bacher, R.F. *Phys. Rev*. **1933**, *43*, 264. (c) Ufford, C.W. *Phys. Rev*. **1933**, *44*, 732.
9. (a) Coester, F. *Nucl. Phys.* **1958**, *7*, 421. (b) Coester, F.; Kümmel, H. *Nucl. Phys*. **1960**, *17*, 477. (c) Čížek, J. *J. Chem. Phys*. **1966**, *45*, 4256.
10. Raghavachari, K.; Trucks, G.W.; Pople, J.A.; Head-Gordon, M. *Chem. Phys. Lett*. **1989**, *157*, 479.
11. Dreuw, A.; Head-Gordon, M. *Chem. Rev.* **2005**, *105*, 4009.
12. Hartree, D.R.; Hartree, W.; Swirles, B. *Phil. Trans. R. Soc. A* **1939**, *738*, 229.

13. (a) Hegarty, D.; Robb, M.A. *Mol. Phys.* **1979**, *38*, 1795. (b) Siegbahn, P.E.M.; Heiberg, A.; Roos, B.O.; Levy, B. *Phys. Scripta* **1980**, *21*, 323.

14. Buenker, R.; Peyerimhoff, S.; Butscher, W. *Mol. Phys.* **1977**, *35*, 771.

15. (a) Andersson, K.; Malmqvist, P.-A.; Roos, B.O.; Sadlej, A.J.; Wolinski, K. *J. Phys. Chem.* **1990**, *94*, 5483. (b) Andersson, K.; Malmqvist, P.-A.; Roos, B.O. *J. Chem. Phys.* **1992**, *96*, 1218.

16. (a) Kowalski, K.; Piecuch, P. *J. Chem. Phys.* **2000**, *113*, 18. (b) Piecuch, P.; Włoch, M. *J. Chem. Phys.* **2005**, *123*, 71.

17. Hohenberg, P.; Kohn, W. *Phys. Rev.* **1964**, *136*, 864.

18. Kohn, W.; Sham, L.J. *Phys. Rev.* **1965**, *140*, 1133.

19. Aiken, H.H. *Proposed automatic calculating machine.* **1937**, 18.

20. Barker, K.J.; Davis, K.; Hoisie, A.; Kerbyson, D.J.; Lang, M.; Pakin, S.; Sancho, J.C. *Supercomputing '08* **2008**, 1.

CHAPTER 2: ENERGY EFFICIENT COMPUTATIONAL CHEMISTRY: COMPARISON OF x86 AND ARM SYSTEMS

A paper published in the *Journal of Chemical Theory and Computation*

Kristopher Keipert, Gaurav Mitra, Vaibhav Sunriyal, Sarom S. Leang, Masha Sosonkina, Allistair Rendell, and Mark S. Gordon

Abstract

The computational efficiency and energy-to-solution of several applications using the GAMESS quantum chemistry suite of codes is evaluated for 32-bit and 64-bit ARM-based computers, and compared to an x86 machine. The x86 system completes all benchmark computations more quickly than either 3ARM system and is the best choice to minimize time to solution. The ARM64 and ARM32 computational performances are similar to each other for Hartree-Fock and density functional theory energy calculations. However, for memory-intensive second order perturbation theory energy and gradient computations the lower ARM32 read/write memory bandwidth results in computation times as much as 86% longer than on the ARM64 system. The ARM32 system is more energy efficient than the x86 and ARM64 CPUs for all benchmarked methods, while the ARM64 CPU is more energy-efficient than the x86 CPU for some core counts and molecular sizes.

Introduction

It is widely recognized that energy usage is a major bottleneck in the pursuit of improving computational performance. This reflects in part the demise of Dennard scaling[1,2], but also fundamental limitations on the energy that can be provided to a single chip regardless of the transistor count. Consequently, computational application developers and users will increasingly need to consider both speed and energy and the interplay between these two metrics. Clear evidence of this trend is seen in the rapid rise of energy-optimized accelerators and co-processors and in the availability of advanced power management facilities on modern processors.

In the pursuit of new energy-efficient hardware designs, low-power mobile computing driven by ARM-based system-on-chip (SoC) processors has aroused significant interest within the high performance computing (HPC) community. These systems are designed with energy efficiency in mind, typically utilizing 32-bit CPUs that are optimized for 32-bit arithmetic. This may be adequate for mobile applications but for quantum chemistry applications large memory and double precision floating point arithmetic is usually required. And while ARM-based devices are now being used in servers it is not yet clear whether either 32-bit ARM-based SoC computers or more recent 64-bit ARM CPUs are viable for HPC workloads. The popular GAMESS[3] quantum chemistry package is used on a wide range of HPC architectures and is therefore a useful test bed for assessing the performance of novel architectures. The present work focuses on measuring performance and energy-to-solution of GAMESS workloads on two ARM-based systems, a 32-bit NVIDIA Jetson TK1 and a 64-bit APM Xgene1 X-C1. The two ARM

systems are also compared to a 64-bit Haswell x86 Intel Xeon-E5 processor. A set of commonly used computational chemistry techniques are evaluated, namely Hartree-Fock (HF) self-consistent field (SCF)[4], density functional theory (DFT)[5-7] and 2nd-order Møller-Plesset (MP2)[8,9] energy and gradient calculations.

Computational Details

The GAMESS performance evaluations used a benchmark set of molecules that contains 99-509 basis functions when using the 6-31G(d)[10-12] basis set. Power measurements were obtained for DFT, HF SCF, MP2 energy, and MP2 gradient calculations. The PBE0 functional[13-15] was used in all DFT calculations. MP2 energy-to-solution and performance comparisons were performed using the distributed data interface (DDI) implementation of MP2[16-18] in GAMESS. In all benchmarks, two-electron integrals were calculated at each (direct) SCF step. The molecules used for benchmarking are listed in *Table 1*. The molecular geometries used for all benchmark calculations were obtained by HF/ cc-PVDZ[19,20] optimizations.

Hardware

The 32-bit ARM machine is an NVIDIA Jetson TK1, configured with a quad-core 2.35 GHz ARM Cortex-A15 CPU (ARMv7-A architecture) paired with 2 GB of LP-DDR3 RAM. The 64-bit ARM machine is an AppliedMicro (APM) X-Gene X-C1 with an 8-core 2.4 Ghz APM883208-X1 CPU (ARMv8-A architecture) and 16 GB of DDR3 memory. The Haswell x86

machine utilizes an 18-core Intel Xeon E5-2699 v3 CPU clocked to the maximum turbo frequency of 3.6 GhZ and 32 GB of DDR4 memory.

To consider the impact on computational performance due to the different system memory types used in each machine, the DRAM read and write bandwidths were measured with the LMBench[21] performance analysis suite. For small memory transactions, 1.05MB in size, the read/write bandwidth is 18.0/12.6 GB/s for the x86 system, 5.9/9.5 GB/s for the ARM64 system, and 4.3/9.2 GB/s for the ARM32 system. For larger memory transactions, 67.11 MB in size, the read/write bandwidth is 10.6/7.6 GB/s for the x86 system, 5.1/9.3 GB/s for the ARM64 system, and 1.2/3.2 GB/s for the ARM32 system. The 32-bit x86 4 GB physical memory capacity limitation is expanded to 1 TB for the 32-bit ARMv7-A architecture via 40-bit physical memory address space support. Also, note that on the 32-bit ARM system double precision numbers are moved between the CPU registers and system memory locations in two 4-byte segments, while on the 64-bit CPUs the entire 8-byte number can be moved to memory with a single instruction.

Software

GAMESS was compiled for the x86 and ARM32 systems with GCC v4.8 and with GCC v5.1 on the ARM64 system (v5.1 is the first version with compiler tuning capabilities for the X-Gene1 CPU). BLAS routines were provided by the ATLAS v3.11 math library[22], natively built for each machine to take advantage of automatic tuning of BLAS routines for each hardware type.

Energy/Power Measurements

The Running Average Power Limit (RAPL)[23] software interface which reads energy consumption information from model-specific registers on an x86 CPU was used to measure the DC power consumption of the 18-core Haswell CPU. RAPL measurements were reported every 0.2 seconds. The DC power consumption of the 64-bit ARM CPU was measured by placing a Fluk i1010 AC/DC current clamp around the wire from the Power Supply Unit (PSU) that supplies power to the CPU. The current clamp was connected to a multimeter which stored current measurements every 0.5 seconds on a remote server. The current used by the ARM32 Jetson system was measured using a uCurrent Gold high precision current measurement tool and an mbed LPC1768 micro-controller with a 12 bit analog-to-digital (ADC) converter, ranging from 0-3.3V. To measure the current, a 0V supply line for the system was routed through the current side of the uCurrent Gold. The ADC was then connected across the voltage output pins of the uCurrent Gold. Serial connections were used to send start and stop signals from the Jetson to the measuring device and to send the measurements from the measuring device to the measuring computer.

The power measurements reported for both the x86 Haswell and ARM64 systems are only for the CPU. The RAPL interface used for measurements of the x86 system provides energy consumption information for the isolated CPU socket. The current clamp used for ARM64 measurements probes the +12V wire from the ATX power supply unit that powers the CPU only. The ARM32 Jetson uses an AC adapter that has a single power supply output. ARM32 energy

measurements are for the entire system and include power consumption for components such as the fan and memory in addition to the CPU.

Results and Discussion

Computational Efficiency

The CPU wall clock times for the various methods on the different platforms are shown in *Figure 1* normalized according to the number of basis functions in each molecule. For all methods employed there is an increase in computational time per basis function as the system size increases. This reflects the worse-than-linear scaling of all methods. For the DFT energy computations in *Figure 1A*, the x86 single core performance is consistently better by a factor of ~3 than both ARM CPUs, with little change in the ratio as the system size increases. The x86 performance relative to ARM64 decreases to a factor of ~2.8 when 8 cores are used. The performance of the ARM32 and ARM64 CPUs are within ±10% of each other.

The results for the HF SCF energy shown in *Figure 1B* are similar to those for the DFT energy. That is, on average the ARM32 computation is 3.3% slower than the ARM64 computation while the ARM64 system takes on average 3.17x/3.16x/2.90x/2.69x longer than the x86 system for HF calculation execution time with 1/2/4/8 cores. For the MP2 energy and gradient calculations memory requirements limit the calculations on the ARM32 system to four molecules that contain 99-250 basis functions, while other restrictions due to the DDI

implementation limits the calculations to six molecules in the range 99-405 basis functions on the ARM64 and x86 systems.

The computation times for the MP2 energy calculations (*Figure 1C*) show the largest difference in performance between ARM32 and ARM64 among all analyzed computation types. Furthermore, the performance degradation of the ARM32 CPU relative to ARM64 worsens with increasing system size and the number of cores used. For example, the MP2 energy computation time for the smallest system, pentane, is 8.6%/13.6%/20.7% greater on 1/2/4 ARM32 cores compared to the same number of ARM64 cores, and 10.0%/22.8%/44.1% greater on 1/2/4 ARM32 cores than the same number of ARM64 cores for the largest molecule that can be run on ARM32 (TNT). By contrast no such correlation is found between the system size or the number of active cores and the relative computational performance when comparing the x86 system to the ARM64 system. On average, the MP2 energy calculations take 3.27x/3.37x/3.32x/3.25x more execution time on the ARM64 system than the x86 system for 1/2/4/8 cores.

For the MP2 gradient (*Figure 1D*), there is a weak correlation between the number of CPU cores used and the relative system performance for ARM64 vs. ARM32, but no such correlation is observed for molecule size. On average, the ARM64 system executes MP2 gradient calculations in 8.5%/10.3%/10.4% less time than the ARM32 system for 1/2/4 cores. The performance benefits of the x86 system relative to the ARM64 system decrease when the number of cores used for the computation is increased. With the exception of the largest molecule (THC: 405 basis functions) the MP2 gradient calculation using the x86 system is on average 2.95x/2.89/2.80/2.67x faster than the ARM64 system with 1/2/4/8 cores. No consistent

correlation between system size and relative performance of x86 vs. ARM64 is observed, but the relative advantage in computational speed for the x86 machine relative to the ARM64 system for the MP2 gradient calculation is greatest for the largest molecule: 3.54x/3.90x/3.61x/3.65x with 1/2/4/8 cores. In general, the ARM32 system performance is worse than the ARM64 system performance for MP2 calculations in contrast to the similar performance observed for the less memory-intensive HF SCF and DFT energy calculations. This degradation in performance may be due to the relatively low read and write bandwidths which were measured for the LPDDR3 RAM of the ARM32 device.

Energy Consumption

Figure 2 shows the energy consumption per basis function for (A) the DFT energy, (B) the HF SCF energy, (C) the MP2 energy, and (D) the MP2 gradient calculations measured for the x86, ARM64, and ARM32 systems. For the DFT calculations averaged over all molecules the ARM32 system requires 31.8%/36.5%/44.3% of the energy consumed by the x86 CPU for 1/2/4 core jobs, while the ARM64 CPU requires 116.2%/102.9%/89.5%/79.5% of the x86 CPU energy for calculations on 1/2/4/8 cores. The HF SCF energy calculations (*Figure 2B*) exhibit similar trends for the x86 and ARM64 CPUs for all core counts; that is the x86 calculation is always slightly more energy efficient for all benchmark molecules on 1 core and always less efficient than the ARM64 CPU on 4 and 8 cores. The ARM32 system consumes an average of 31.1% of the x86 CPU energy for 1 core, 36.3% for 2 cores, and 48.6% for 4 cores.

The MP2 energy efficiency results are shown in *Figure 1C*. The ARM64 calculations on average and using 1/2 cores consumes 29.1%/11.1% more energy than 1/2 x86 cores; when using 4 or 8 cores the x86 machine falls within ±3% of the analogous results obtained on the ARM64 machine. The ARM32 system is the most energy efficient, but this energy efficiency rapidly diminishes when increasing the number of cores. On 1/2/4 cores the MP2 energy calculations on the ARM32 system and averaged over all molecules uses 36.5%/48.4%/71.8% of the energy required for the equivalent calculations on the x86 system. For the MP2 gradient computations, the energy consumption of the ARM64 CPU averaged over all molecules is 117%/102%/90%/86% of the x86 CPU energy used for the same computations on 1/2/4/8 cores. On the ARM32 system the 99-250 basis function computations consume on average 31.9%/40.0%/51.0% of the energy used by the x86 CPU for 1/2/4 cores, similar to the relative energy consumption for the DFT energy and HF SCF computations.

Busy/Idle Core Energy Usage

When running a calculation on less than the total number of CPU cores, the unused cores consume energy in the idle state. To examine the efficiency of running parallel versus multiple copies of sequential code, and in order to estimate the energy consumed by busy and idle cores, the energy usage was measured for MP2 gradient calculations on TNT performed using varying levels of CPU core saturation. The energies and times used per basis function are shown in *Table 2*. The 1-core values correspond to single 1-core computations while all remaining cores are idle. The 8-core values correspond to 8 cores used for a single computation running in parallel. This

fully saturates the available ARM64 cores but leaves 10 idle cores for the x86 CPU. Also shown is the energy usage for running 8x1-core jobs simultaneously.

The 8x1-core parallel and 8-core schemes have similar energy consumption and calculation times for all computation steps for both x86 and ARM64 CPUs with the exception of the MP2 energy on the x86 system. That is, it is as efficient to run 8 identical single core calculations simultaneously as it is to run one calculation in parallel using 8 cores, and then to repeat that calculation 8 times. For the x86 MP2 energy there is a slight difference: the 8x1 core parallel scheme consumes 9.2% more energy and takes 8.3% more execution time per basis function compared to the 8-core computation. Overall the results suggest that the HF SCF, MP2 energy, and MP2 gradient algorithms in GAMESS do not have significant computational cost overhead for parallel task coordination. Also, there is no significant off-chip memory or I/O contention when running 8 compute processes in parallel.

To calculate the power consumption of individual busy and idle cores their energy usage is approximated using Eqs 1 and 2, respectively. $Core_{Max}$ is the number of physical cores per CPU: 18 for x86 and 8 for ARM64. In Eq 1, the "saturated" subscript indicates the value for $Core_{Max}$ jobs running simultaneously, each using one core. This corresponds to the "x86 18x1-Core, Parallel" and "ARM64 8x1-Core, Parallel" values (Table 2). In Eq 2, the "*n-core"* subscript indicates the value for a single job running on n cores. The value n=1 is chosen for idle core calculations in this study and corresponds to the "x86 1-Core" and "ARM64 1-core" values in *Table 2*.

$$Busy\ Core\ Power = \frac{Energy_{Saturated} / Time_{Saturated}}{Core_{Max}} \quad (1)$$

$$Core\ Power = \frac{Energy_{n\text{-}Core} / Time_{n\text{-}Core} - (n\ *\ Busy\ Core\ Power)}{Core_{Max} - n} \quad (2)$$

The calculated power consumption per *busy* core during the HF SCF/MP2 energy/MP2 gradient calculations is 7.93W/7.65W/6.74W for the x86 CPU and 3.35W/3.64W/3.53W for the ARM64 CPU. The calculated power consumption per *idle* core during the HF SCF/MP2 energy/MP2 gradient calculations is 2.47W/2.57W/2.55W for the x86 CPU and 2.62W/2.21W/2.10W for the ARM64 CPU. Extrapolating the average idle core power consumption during the three calculation types to the $Core_{Max}$ value, the *calculated* total power consumption for an idle CPU is 45.57W for the x86 CPU and 18.46W for the ARM64 CPU.

For comparison power usage was measured *experimentally* for both CPUs in the idle state over a period of 1 hour. It was found that while on the ARM64 the average measured value of 19.10W agreed well with the derived value of 18.46W, the measured value of 16.83W on the x86 CPU is significantly less than the derived value of 45.57W. This 2.7x reduction in power usage presumably reflects the fact that the Haswell x86 CPU includes the C7 sleep state feature to lower idle core power consumption when the entire CPU is idle. In terms of ideal energy efficiency for the quantum chemistry algorithms analyzed, the results clearly demonstrate that it is much more important to saturate all available cores regardless of the number of cores per

computation than it is to choose between parallel and back-to-back serial computation executions. This is particularly true for the Haswell architecture, which incurs a relatively large incremental energy cost when left in the completely idle CPU state. This is not observed for the ARM64 CPU.

Power Trace

To explore whether energy usage changes significantly during the course of the calculations *Figure 3* shows a trace of the instantaneous power consumption of the x86 and ARM64 CPUs and ARM32 system during an MP2 gradient calculation on TNT running on 4 CPU cores. The average idle energy consumption over a 1-hour measurement is plotted in *Figure 3* for the x86, ARM64 and ARM32 systems, indicated by times from -100 to 0 seconds. The average x86 idle CPU power consumption of 16.83W is initially lower than the 19.10W average of the ARM64 CPU, but within 1 second of the HF SCF calculation, power consumption increases by 71.95W for the x86 CPU, but only to 23.12W for the ARM64 CPU. The ARM32 system uses less power than either ARM64 or x86, with an average idle power consumption of 3.21W which increases to 10.58W after 1.0 second has elapsed in the HF SCF calculation.

Table 3 shows the mean, standard, and relative standard deviations of the x86, ARM64 and ARM32 systems during the CPU power trace calculation. On all machines, once the computation has begun, fluctuations in power usage are relatively small. For the x86 and ARM64 CPUs, the mean power consumption is highest for the MP2 energy calculation, followed by the MP2 gradient and the HF SCF calculations. The ARM32 MP2 gradient calculation

consumes slightly more power in the gradient step, followed by the HF SCF energy and the MP2 energy calculation. The standard deviation of CPU power consumption is highest for the x86 CPU for each calculation step of the power trace at 1.57W for the HF SCF step, 2.17W for the MP2 energy step, and 2.52W for the MP2 gradient step. The relative standard deviation, which takes the magnitude of the average power consumption into account, is lowest for the x86 CPU at 2.52% for the HF SCF step, 2.17% for the MP2 energy step, and 1.57% for the MP2 gradient step. The ARM64 CPU power consumption is the most consistent between calculation steps with a standard deviation of 0.86W for the HF SCF step, 0.98W for the MP2 energy step, and 0.81W for the MP2 gradient step.

Conclusions

Supercomputers capable of exascale level computations will greatly extend the complexity of feasibly solvable problems in computational sciences. The most significant barrier to exascale supercomputers is the relatively poor energy efficiency of modern computer hardware. To reach the exascale it is therefore imperative that improvements in CPU technology address both computational throughput and energy efficiency. This work has explored these issues in the context of a widely used quantum chemistry package running on ARM32, ARM64 and x86 processors. For all methods and molecules considered the x86 CPU is the clear choice in terms of minimizing time to solution. For energy efficiency the ARM32 system offers the best performance, but the 32-bit architecture limits the utility of this system for quantum chemistry calculations. While these limitations have been lifted with the advent of ARM64 systems, it appears that this has come at a significant cost to energy usage without a significant increase in

performance. Whether the latter is in part a reflection on the immaturity of the ARM64 compiler and runtime remains to be seen.

Acknowledgements

The authors thank Intel Corp. and NVIDIA Corp. for their support of this work. K.K., V.S., S.L., M.S., and M.S.G. thank the Air Force Office of Scientific Research for their support of this work under AFOSR Award No. FA9550-12-1-0476.

References

1. Dennard, R. H.; Gaensslen, F.; Yu, H-N.; Rideout, L.; Bassous, E.; LeBlanc, A. *IEEE J. Solid-State Circuits* **1974**, *SC-9*, 256.
2. The impact of Dennard's scaling theory. *IEEE Solid-State Circuits Society News* **2007**, *12 (1)*.
3. Gordon, M. S.; Schmidt, M. W. *In theory and applications of computational chemistry: The first forty years*; Elsevier: Amsterdam, The Netherlands, **2005**.
4. Szabo, A.; Ostlund, N. S. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*; Dover Publications, Mineola, NY, **1996**.
5. Hohenberg, P.; Kohn, W. *Phys. Rev. B.* **1964**, *136*, 864.
6. Kohn, W.; Sham, J. *J. Phys. Rev. A* **1965**, *140*, 1133.
7. Parr, R.; Yang, W. *Density-Functional Theory of Atoms and Molecules*; International Series of Monographs on Chemistry; Oxford University Press, New York, **1989**.

8. Møller, C.; Plesset, M. S. *Phys. Rev.* **1934**, *46*, 618.

9. Bartlett, R. J. *Annu. Rev. Phys. Chem.* **1981**, *32*, 359.

10. Ditchfield, R.; Hehre, W. J.; Pople, J. A.. *J. Chem. Phys.* **1971**, *54*, 724.

11. Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56*, 2257.

12. Rassolov, V. A.; Ratner, M. A.; Pople, J. A.; Redfern, P. C.; Curtiss, L. A. *J. Comput. Chem.* **2001**, *22*, 976.

13. Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.* **1996**, *77*, 3865.

14. Perdew, J. P.; Burke, K.; Ernzerhof, M. *Phys. Rev. Lett.* **1997**, *78*, 1396.

15. Adamo, C.; Barone, V. *J. Chem. Phys.* **1999**, *110*, 6158.

16. Fletcher, G. D.; Schmidt, M. W.; Bode, B. M.; Gordon, M. S *Comput. Phys. Commun.* **2000**, *128*, 190.

17. Olson, R.; Schmidt, M.; Gordon, M.S.; Rendell, A. *Supercomputing 03'* **2003**, 41.

18. Federov, D. G.; Olson, R. M.; Kitaura, K.; Gordon, M. S.; Koseki, S. *J. Comput. Chem.* **2004**, *25*, 872.

19. Dunning, Jr., T. H. *J. Chem. Phys.* **1989**, *90*, 1007.

20. Woon, D. E.; Dunning, Jr., T. H. *J. Chem. Phys.* **1995**, *103*, 4572.

21. McVoy, L.; Staelin, C. *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference* **1996**, 23.

22. Whaley, R. C.; Petitet, A.; Dongarra, J. *J. Parallel Computing*, **2000**, *27*, 3.

23. David, H.; Gorbatov, E.; Hanebutte, Ulf R.; Khanna, R.; Le, C. In *ACM/IEEE International Symposium on Low-Power Electronics and Design* **2010**, 189.

Table 1. Benchmark molecule specifications

Molecule	**Chemical Formula**	**Number of Basis Functions**
Pentane	C_5H_{12}	99
Asparagine	$C_4H_8N_2O_3$	151
Nicotine	$C_{10}H_{14}N_2$	208
Trinitrotoluene (TNT)	$C_7H_5N_3O_6$	250
Indigo	$C_{16}H_{10}N_2O_2$	320
Tetrahydrocannabinol (THC)	$C_{21}H_{30}O_2$	405
Adenosine Triphosphate (ATP)	$C_{10}H_{16}N_5O_{13}P_3$	509

Table 2. Energy consumption and computation time per basis function of x86 and ARM64 CPUs for TNT (250 basis functions) MP2 gradient calculation steps for 1-core and 8-core calculations, and for 8 and 18 1-core calculations in parallel

		Energy/B. Function, J	**Time/B. Function, s**
HF SCF Energy	x86 1-Core	19.849	0.397
	x86 8-Core	5.658	0.062
	x86 8x1-Core, Parallel	5.665	0.063
	x86 18x1-Core, Parallel	4.139	0.029
	ARM64 1-Core, Serial	21.676	1.144
	ARM64 8-Core	4.003	0.158
	ARM64 8x1-Core, Parallel	4.155	0.155
MP2 Energy	x86 1-Core	44.784	0.871
	x86 8-Core	13.362	0.145
	x86 8x1-Core, Parallel	14.597	0.157
	x86 18x1-Core, Parallel	13.352	0.097
	ARM64 1-Core	57.932	3.040
	ARM64 8-Core	13.435	0.480
	ARM64 8x1-Core, Parallel	13.855	0.476
MP2 Gradient	x86 1-Core	22.217	0.444
	x86 8-Core	6.190	0.069
	x86 8x1-Core, Parallel	6.133	0.068
	x86 18x1-Core, Parallel	3.884	0.032
	ARM64 1-Core	25.247	1.384
	ARM64 8-Core	5.132	0.188
	ARM64 8x1-Core, Parallel	5.281	0.187

Table 3. Mean, standard deviation, and relative standard deviation of instantaneous power consumption during 250 basis function MP2 gradient calculation for ARM64 and x86 CPU, ARM32 system

	Mean, W	**Standard Deviation, W**	**Relative Standard Deviation, %**
x86			
Hartree-Fock SCF Energy	72.15	2.52	3.50
MP2 Energy	74.58	2.17	2.91
MP2 Gradient	72.98	1.57	2.15
ARM64			
Hartree-Fock SCF Energy	21.57	0.86	3.99
MP2 Energy	22.96	0.98	4.26
MP2 Gradient	22.56	0.81	3.60
ARM32			
Hartree-Fock SCF Energy	11.31	0.97	8.58
MP2 Energy	10.61	1.71	16.09
MP2 Gradient	11.96	0.84	7.06

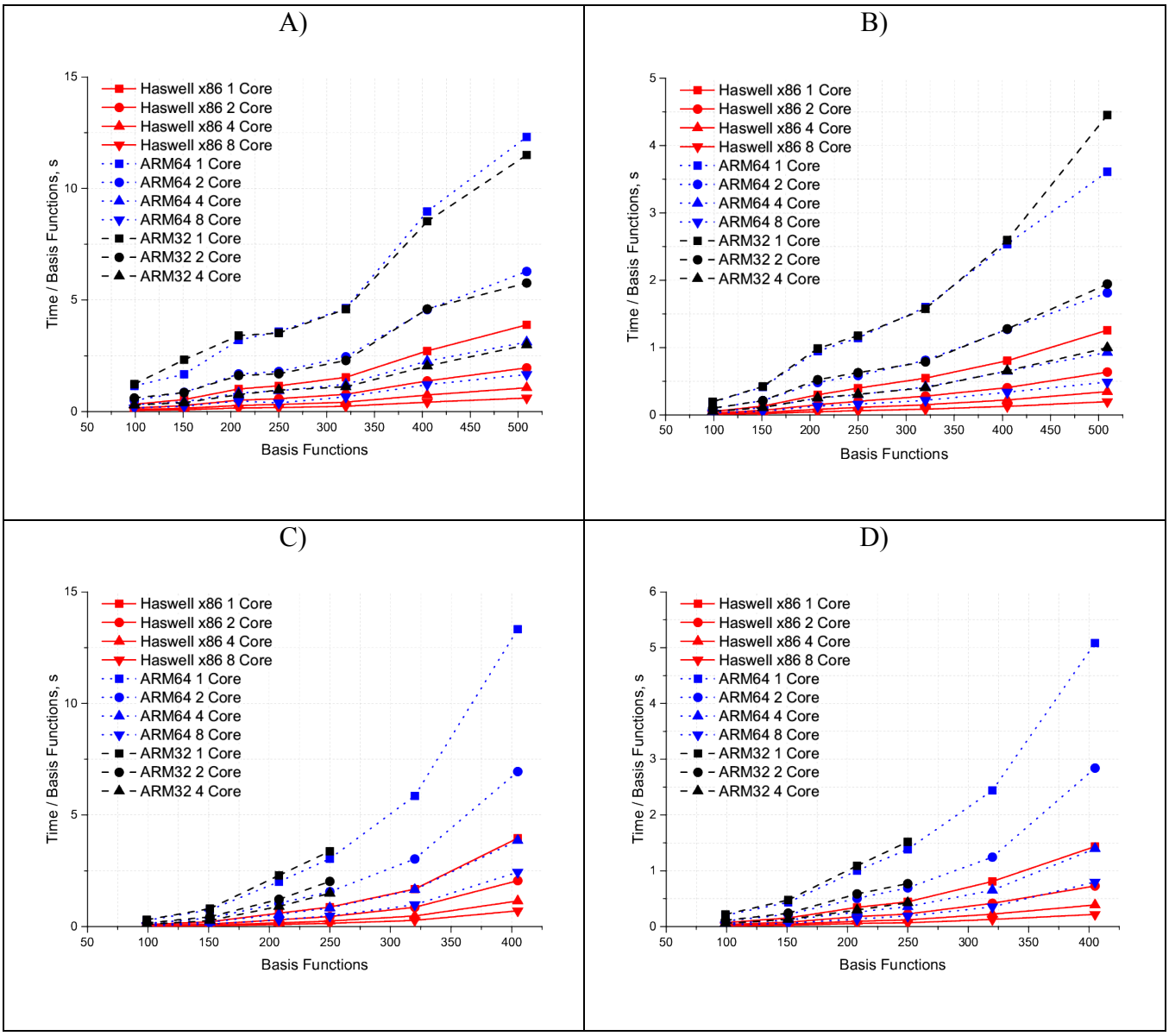


Figure 1. Computation times per basis function for A) DFT energy, B) HF SCF energy, C) MP2 energy, and D) MP2 gradient benchmark calculations

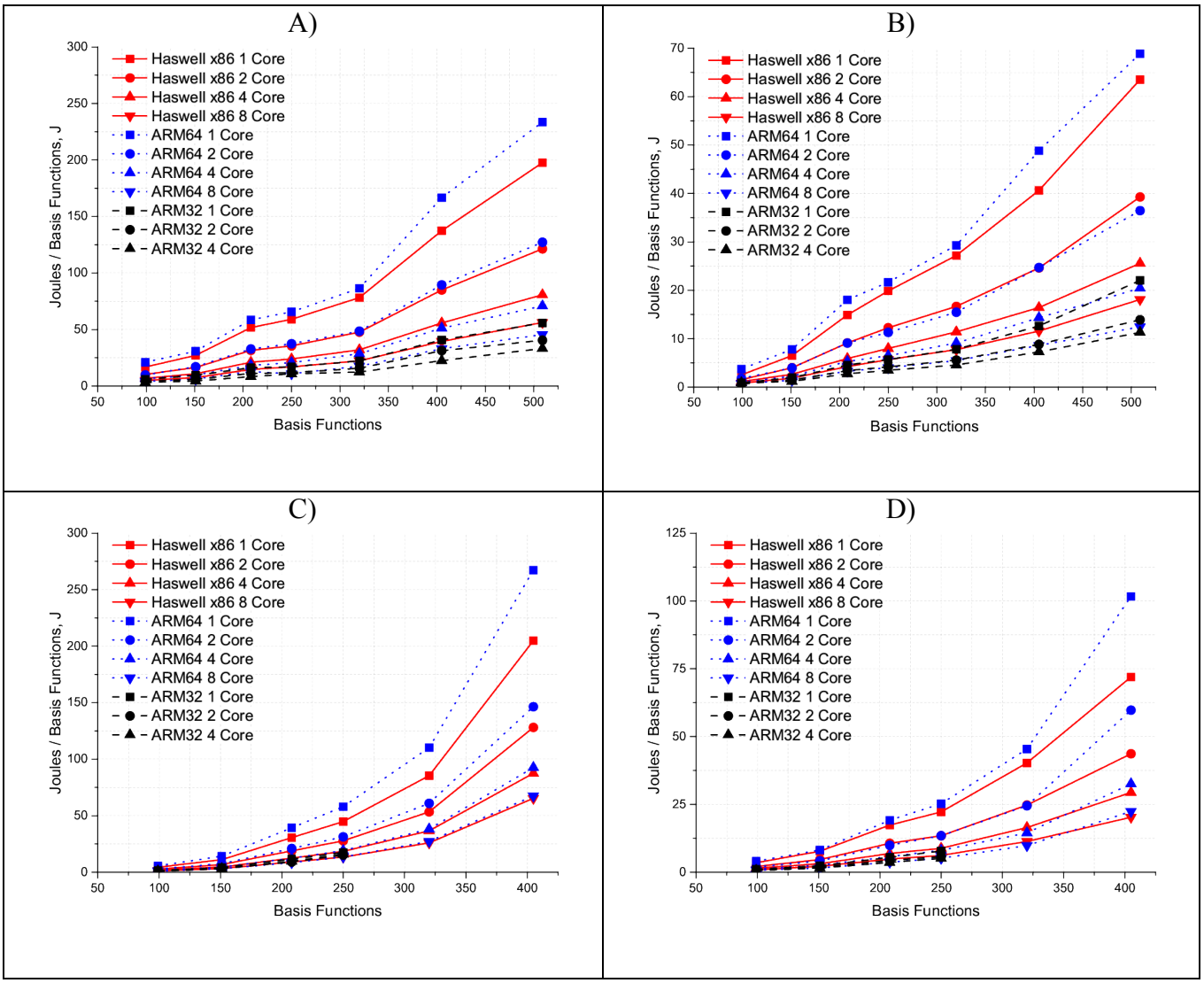


Figure 2. Energy consumption per basis function for A) DFT energy, B) HF SCF energy, C) MP2 energy, and D) MP2 Gradient benchmark calculations

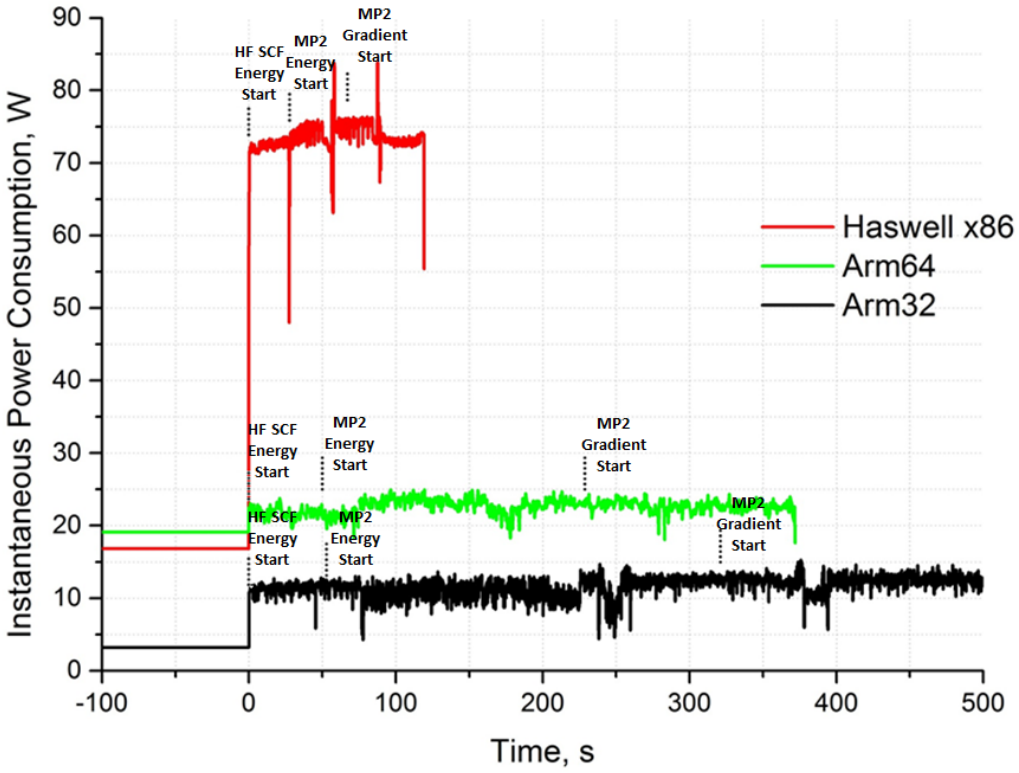


Figure 3. Power trace during TNT MP2 gradient calculation for ARM64 and x86 CPU, ARM32 systems, 4 CPU cores

CHAPTER 3: PERFORMANCE AND ENERGY EFFICIENCY ANALYSIS OF 64-BIT ARM USING GAMESS

A paper published in the *Proceedings of the 2nd International Workshop on Hardware-Software Co-Design for High Performance Computing, Co-HPC 2015*

Ananta Tiwari, Kristopher Keipert, Adam Jundt, Joshua Peraza, Sarom S. Leang, Michael Laurenzano, Mark S. Gordon, and Laura Carrington

Abstract

Power efficiency is one of the key challenges facing the HPC co-design community, sparking interest in the ARM processor architecture as a low-power high-efficiency alternative to the high-powered systems that dominate today. Recent advances in the ARM architecture, including the introduction of 64-bit support, have only fueled more interest in ARM. While ARM-based clusters have proven to be useful for data server applications, their viability for HPC applications requires an in-depth analysis of on-node and inter-node performance. To that end, as a co-design exercise, the viability of a commercially available 64-bit ARM cluster is investigated in terms of performance and energy efficiency with the widely used quantum chemistry package GAMESS. The performance and energy efficiency metrics are also compared to a conventional x86 Intel Ivy Bridge system. A 2:1 Moonshot core to Ivy Bridge core performance ratio is observed for the GAMESS calculation types considered. Doubling the number of cores to complete the execution faster on the 64-bit ARM cluster leads to better energy efficiency

compared to the Ivy Bridge system; i.e., a 32-core execution of GAMESS calculation has approximately the same performance and better energy-to-solution than a 16-core execution of the same calculation on the Ivy Bridge system.

Introduction

The energy consumption of large scale high performance computing (HPC) systems is becoming an increasing concern as the computational science community heads towards the exascale era. Current estimates indicate that processor efficiencies will have to evolve from the current 5 GFLOPS/Watt to 50 GFLOPS/Watt for exascale machines to be viable. While multiple processor architectures are being considered in the pursuit of more energy-efficient HPC, it is almost certain that the ARM architecture will figure prominently into the solution set, as evidenced by its presence in the Mont-Blanc project[30]. Furthermore, the entire HPC market is only a small fraction of the total computing market and, therefore, relies primarily on commodity processor technology. In mobile computing, the ARM architecture plays a large role as the most ubiquitous energy-efficient processor architecture available. Consequently, investors and companies have recognized the potential of the ARM architecture in HPC and enterprise computing. For example, HP, Dell, and others are fielding enterprise-class servers based on ARM processors. NVIDIA has developed a hybrid ARM-GPGPU system on chip (SoC) which was initially deployed at the Barcelona Supercomputing Center for the Mont-Blanc project.

Given the rapid penetration of ARM systems into the HPC market, it is critical that the performance and energy efficiency of ARM is understood in the context of well-established

architectures used in HPC. Towards that end, the performance, parallel efficiency, and power efficiency of the modern ARM-based HP Moonshot system is analyzed, and compared to an Intel Ivy Bridge system. As a co-design exercise, this study is conducted for the prominent large scale computational chemistry package GAMESS[32]. Of primary interest is the investigation of which types of computations within the GAMESS package see the best performance scaling and energy-efficiency on the Moonshot system, in addition to the identification of architectural features that bottleneck the performance. The Intel Ivy Bridge system is used as a reference architecture for the bottleneck analysis. The paper is organized as follows: First, descriptions are provided for the GAMESS software package and for each type of GAMESS calculation explored in this study. Next, the methodology used for system benchmarking is described. This description includes details about how the metrics relevant to the performance and energy efficiency of the target systems are measured. Next, the results which demonstrate the scalability and energy efficiency of various types of chemistry calculations on the Moonshot system are presented. The Moonshot system is then compared to Ivy Bridge using the same set of metrics. Related work is then summarized, followed by concluding remarks.

Background – GAMESS

The General Atomic and Molecular Electronic Structure System (GAMESS) is a general purpose electronic structure code, with a primary focus on *ab initio* quantum chemistry calculations. GAMESS is used on a wide range of processor technologies all over the world and, as such, presents itself as a useful application to assess the performance and energy consumption aspects of upcoming processor architectures. The computation parallelization in GAMESS is

achieved using the DDI (Distributed Data Interface) library[14,28]. In GAMESS, for each process that computes chemistry calculations, there is an associated "data server" process that services data requests from the distributed arrays. A 16-core run of GAMESS will, for example, have 8 compute processes and 8 data servers. The following specific GAMESS calculations are considered in this work, using the CCD basis set[10]:

Fragment Molecular Orbital (FMO)

In FMO, user specified parameters partition a molecular system, which breaks up the computational workload into chunks which are distributed based on the number of compute nodes available. FMO allows for the use of highly accurate *ab initio* quantum chemistry methods on large molecular systems. For this benchmark, FMO is used to calculate Hartree-Fock (HF) SCF energies. The molecular systems investigated are clusters of 32 and 64 water molecules.

Hartree-Fock, Second Order SCF (HF-SOSCF)

Hartree-Fock energies are computed for several benchmark systems. Second order orbital optimizations are performed. Both DISK and DIRECT options are evaluated. For DIRECT runs, integrals are recomputed at each SCF iteration instead of reading them from disk. DISK runs retrieve the stored integrals from disk. Three benchmark molecules are considered for the HF-SOSCF calculations: silatrane ($C_7H_{17}O_4NSi$), nicotine ($C_{10}H_{14}N_2$), and trinitrotoluene (TNT) ($C_7H_5N_3O_6$).

Configuration Interaction with Single Excitations (CIS)

A CIS energy computation of the porphin molecule ($C_{20}H_{14}N_4$) is analyzed. CIS is an excited state calculation, for which the computational scaling depends on both system size and the number of excited states calculated. A standard HF calculation is first performed to obtain a

reference wavefunction, which is used in a linear combination with configuration state functions representing single electron excitations while neglecting orbital optimization.

2nd Order Møller-Plesset Perturbation (MP2), and Resolution-of-Identity Variant (RI-MP2)

MP2/RI-MP2[17] are second-order perturbation theory energy calculations. In these post Hartree-Fock methods, electron correlation effects are added via a perturbative correction. The MP2 energy calculation is more demanding on both CPU and memory compared to both Hartree-Fock and CIS computations. The RI-MP2[36] approximation can be as much as 40 times faster than MP2 and is nearly as accurate. Notably, the RI-MP2 approximation employs significant use of matrix-matrix operations in comparison to the domination of vector-vector operations in HF and standard MP2 calculations. The TNT molecule is used in evaluation of the MP2 and RI-MP2 energy calculations.

Methodology

Platforms

The Moonshot system consists of 128 cores in a multi-node server configuration. The system has 16 nodes (also called cartridges) connected via 10GigE interconnect. Each node has 64 GB memory (DDR3-1600) and one AppliedMicro® X-GeneTM 1 processor, which consists of 8 64-bit ARM cores, each of which is clocked at 2.4 GHz. Each core has 32KB L1 data cache and 32KB L1 instruction cache. Each core pair shares a 256KB L2 cache and all 8 cores in a

node share 8MB L3 cache. Each node is configured with a solid state drive (SSD) for storage with 120 GB capacity. In addition to the scaling runs on the Moonshot with multiple core counts, this paper compares the performance and energy efficiency of the Moonshot system to a dual-socket Intel® Ivy Bridge node with 32 GB memory (DDR3-1333).

Each socket consists of 8 cores, each of which is clocked at 2.6 GHz. Each core has 32KB L1 data cache, 32KB L1 instruction cache and 256KB L2 cache. 8 cores in a socket share 20MB L3 cache. The system has traditional disks running at 7200 RPM. It was discovered that the performance of some of the benchmark calculations are highly sensitive to the speed of the storage sub-system, so an SSD with 120 GB capacity was added to the Ivy Bridge system. Unless otherwise noted, all of the relevant results presented in this paper are obtained using the SSD drive.

Iperf[12] is used to measure the network bandwidth between nodes on the Moonshot and between sockets on the Ivy Bridge system. The tool saturates the link between the nodes/sockets and determines the maximum bandwidth between nodes/sockets. Inter-node bandwidth on the Moonshot system is 9.89 Gbps (limited by the 10 Gbps switch that connects the nodes of the system) and 30.4 Gbps between sockets on the Ivy Bridge.

GAMESS Compilation Environment

GAMESS is compiled on the Moonshot system using gcc-5.1.0 with the flags “-O2 -mcpu=xgene1 -fno-aggressive-loop-optimizations -march=armv8-a -mtune=xgene1”. Same

compiler version and analogous optimization flags ("-O2 -march=IvyBridge -mtune=intel -fno-aggressive-loop-optimizations") are used to compile GAMESS on the Ivy Bridge system. Both systems use ATLAS[37] version 3.11.34 for BLAS.

Power Measurement

In addition to performance at different scales, energy consumption is also of interest. Only the dynamic power draw is measured. To accomplish this, the idle power of each system is measured first. The idle power draw is then subtracted from the total power drawn during application runs. Specific methods used to measure the idle power and the active power for each of the systems are described below.

Moonshot

The Moonshot power measurements rely on the HP iLO (integrated Lights-Out) server management technology[1]. iLO allows total power draw measurement at the chassis level, as well as power draw measurement at the cartridge level. The chassis level power, which can be measured at one-second granularity, includes the power drawn by heavy duty chassis fans. Rapid changes in the rotational speeds of these fans make the power draw measurement at the chassis level noisy and unreliable. Therefore, power measurements are obtained at the per-cartridge level. The per-cartridge power is measured once every 15 seconds, and GAMESS is executed multiple times for a period of at least 3 minutes to obtain a reliable measurement. Power drawn by the networking components can also be measured using the iLO interface. Networking components draw a constant power of approximately 50 watts. To calculate dynamic power, the

idle power is first measured on each of the Moonshot nodes for a period of 30 minutes. The total dynamic power for a scaling run that uses n nodes is calculated using the following formula:

$$P_{Dyn} = (\sum_{i=1}^{n} (L_{P_i} - I_{P_i})) + \frac{N_P}{n} \quad (1)$$

In *Equation 1*, L_{P_i} is the on-load power drawn by node *i*, I_{P_i} is the idle power drawn by node *i*, and N_P is the constant network power. The second term in the equation adds a proportional network power to the dynamic power (P_{Dyn}) calculation. Each run produces a series of series of P_{Dyn} measurements, which are then averaged to get a power draw value for the run.

Ivy Bridge

Power measurements on the Ivy Bridge are taken using a Wattsup[2] device which provides total system power measurement at one second granularity. The device provides a USB interface to obtain the measurements, and this interface is queried from a separate system to ensure that the measurement tools do not increase power draw or slow down the application. The system fans are set to manual control in the BIOS and their power draw is lumped into the idle power measurement so that their effect on dynamic power draw is minimized. Idle power is measured for 30 minutes and P_{Dyn} is calculated by deducting idle power from on-load power.

Metrics

Power, performance, and energy consumption are measured for each of the GAMESS calculations considered for this work. On the Moonshot system, these calculations were run using 8, 16, 32, 64 and 128 cores; on the Ivy Bridge system, the calculations were run using 8

and 16 cores. For scaling studies on Moonshot, an 8-core single node run is taken as the reference, and all measurements are normalized using the measurements in the reference run. A parallel efficiency metric, defined as:

$$P_{e_n} = \frac{T_1}{n * T_n} \quad (2)$$

is also measured. In *Equation 2*, P_{e_n} is the parallel efficiency metric for n nodes (i.e., n × 8 cores), T_1 is the execution time on one node (8 cores) and T_n is the execution time on n nodes. A P_{e_n} metric equal to 1 indicates perfect scaling. For cross-architecture comparisons, two sets of results are presented—one that takes a single socket 8-core run on the Ivy Bridge system as the reference and normalizes the measurements across different core counts on the Moonshot using the reference measurements, and another that takes the dual-socket 16-core run on the Ivy Bridge system as the reference for normalization. In addition to power, performance and energy, the Energy-Delay Product (EDP) is calculated as (energy × performance). The EDP metric emphasizes performance and is widely used in comparing the efficiency of different high-end systems[15].

Performance Analysis Tools

To analyze performance on the Intel system, a suite of tools developed on top of a binary instrumentation toolkit, PEBIL[23], was used. These tools combine static binary analysis information (e.g., approximate structure of the program in terms of functions and loops, and operations within those structures) with dynamic analyses (e.g., basic block counts and cache simulation) to provide an in-depth description of the performance related characteristics of

applications. EPAX[11] was used to analyze the Moonshot system. EPAX provides static binary analyses for both 32-bit and 64-bit ARM architectures. A set of tools that leverages the EPAX static analysis information has also been used to, for example, generate an instruction mix for different structures in the application.

To analyze multi-node runs, the PSiNSTracer tool[35], a light-weight tool that captures communication and computation profiles of MPI applications, was used. The tool intercepts all the MPI calls and collects time spent on each of those calls. The portion of the application execution time that is not attributed to the MPI events is categorized as computation time. This paper analyzes the communication versus computation behavior for only the compute processes. Data servers perform only data-servicing tasks.

Moonshot Performance and Energy Results

This section describes performance scaling and energy efficiency results obtained on the Moonshot system. Each metric is collected at five core-counts on the Moonshot system: 8, 16, 32, 64 and 128. To reduce noise, each data point is measured five times and the average is reported. For each of the GAMESS calculations considered in the paper, the single-node 8-core run is taken as the reference; measurements across other core counts on the Moonshot systems are normalized using the reference measurements. In *Figure 1*, the parallel efficiency for each calculation is shown. Values closer to 1 indicate better scaling. In *Figure 1* the parallel efficiency for each calculation is shown. Values closer to 1 indicate better scaling. In *Figure 2*, the energy

consumption normalized by the single node case is presented. Normalized EDP values are shown in *Figure 3*. Lower values indicate greater efficiency.

FMO Results

FMO calculations show near linear scaling on Moonshot. For a 128-core run with 64 water molecules, the speedup with respect to the 8-core run is 15.9 (linear speedup would be 16). As mentioned earlier, the implementation of FMO breaks the computational workload into chunks, which are then distributed to available compute nodes. The communication between the nodes is minimal, which explains the scaling behavior. Communication-computation profiles reveal that for 128-core runs with 64 waters, on average, less than 5% of the time spent by compute processes is attributed to communication. Normalized EDP (as shown in *Figure 3*) improves the most for FMO calculations at higher core counts on the Moonshot system compared to other calculations. Results for the 32 water system are similar to those for the 64 water system.

HF-SOSCF Results

Recall from the introduction that both disk and direct variants of HF-SOSCF calculations are considered. The relative performance of the two methods (disk versus direct) was investigated using the silatrane molecule, in addition to analysis of the performance scaling, parallel efficiency and EDP. The normalized execution times for the direct method HFSOSCF calculations are shown in *Figure 4*. The execution times for the direct method are normalized to the execution times for the disk method. As demonstrated in the graph, disk based calculations are faster at all core counts. However, the edge that disk based calculations has on direct calculations at 8-core runs diminishes significantly at 128-core runs. Next, the parallel efficiency metric is considered. It is shown in *Figure 1* that disk based methods rank lowest in terms of

parallel efficiency. Parallel efficiency for the silatrane molecule using disk is at 0.4 (at 128 cores) versus 0.74 for the direct method[3]. Normalized EDP points to the same conclusion. The normalized EDP for disk based calculations stops declining beyond 64-core count runs (*Figure 3*); i.e., efficiency stops improving beyond 64 cores. To further investigate the relative difference in scaling between the direct and disk based methods, consider the communication and computation profiles for these methods. Profiles for both the direct and disk methods for the silatrane molecule calculations executed using all 128 cores (16 nodes) on the Moonshot system are shown in *Figure 5*. Poor parallel efficiency with the disk based method can be primarily attributed to the greater communication intensity of this method compared to the direct method. For 128 core direct runs, time spent in communication by compute processes of GAMESS is 11%, compared to 35% for disk runs. Most of this communication time is spent in MPI broadcast events.

CIS Results

The HF Configuration Interaction-Singles (CIS) calculation using the porphin molecule ranks second after the FMO method in parallel efficiency (0.79, see *Figure 1*). The normalized EDP metric indicates higher efficiency at higher core count runs. On average, GAMESS compute processes are engaged in communication events for 13% of the run time for 128-core runs.

[3] Parallel efficiency quantifies what proportion of theoretical maximum speedup is achieved by GAMESS when scaled to multiple nodes on the Moonshot. 0.74 means 74% of the achievable speedup was attained.

MP2/RI-MP2 Results

These calculations show greater parallel efficiency than HF disk based methods but lower efficiency than FMO calculations. Both MP2 and RI-MP2 show identical parallel efficiencies of 0.68 for 128 core runs (*Figure 1*). The normalized EDP metric, which emphasizes performance, continues to decline (*Figure 3*) for large core counts (i.e., strong scaling will continue beyond 128-core runs). Analyses of the computation-communication profile for 128-core runs reveal that compute processes in MP2 calculations, on average, are engaged in communication events for 19% of the run time, compared to 20% of RI-MP2.

Cross-Architectural Study Results

To make cross-architectural performance and energy comparisons, the playing field has been made as level as practicably possible, by using the exact same compilation environment on the two systems. In the preliminary analysis of the performance differences between the two systems, it was discovered that the Moonshot system performed relatively better (even for the same core-counts) than the Ivy Bridge system for HF-SOSCF disk-based calculations. An I/O profiling analysis (performed with the I/O tracer built on top of PEBIL[27]) revealed that the Ivy Bridge system, which was utilizing the traditional spinning hard-disk drive, was spending considerable time on I/O calls. An SSD with similar performance specifications to that of the SSD on the Moonshot was added to the Ivy Bridge system, and that addition improved the performance of HF-SOSCF disk based calculations by up to 2.2×. This paper reports the performance for HF disk based calculations using the SSD on the Ivy Bridge system. The performance, energy and EDP metrics obtained on the Moonshot system across multiple core

counts were normalized to the same metrics obtained on single-socket (8-core) and dual-socket (16-core) runs on the Ivy Bridge system. These comparisons facilitate the investigation of whether using more low-power simpler cores can be beneficial in terms of energy efficiency compared to using relatively few heavy-duty cores.

Single Socket Ivy Bridge

Comparative results are shown in *Figures* 6, *7*, and *8* with 8-core Ivy Bridge metrics used as references. In *Figure 6*, the normalized performance across multiple core count runs on the Moonshot system is shown. Performance on the Moonshot using 8 cores is 1.6× to 2× slower than the 8 core runs on Ivy Bridge. All 8-core runs on the Moonshot use less energy than the corresponding 8-core runs on Ivy Bridge (63% to 77% of the energy needed to run on Ivy Bridge). The normalized EDP metric for 8-core runs, however, show that the Ivy Bridge system is more efficient at running 8-core executions. As the Moonshot scales to 16 cores, the performance in all cases exceeds the performance on 8 cores of the Ivy Bridge. The normalized EDP metric also suggests that greater efficiency can be achieved with large core count runs on the Moonshot. The next natural research questions to ask are—What drives the performance differences between 8-core runs on Moonshot and Ivy Bridge, and what architectural components on the Moonshot tend to bottleneck its performance?

Architectural Bottlenecks

The Moonshot system exposes a set of performance hardware counters that can be used to measure the interactions of software with key on-node architectural components of Moonshot—floating point units, caches, memory, and the branch predictor. These counters can

be measured using PAPI[25], which has support for the X-Gene 1 architecture. The overall idea is to investigate which of these counters correlate with the relative performance difference between 8-core runs on Moonshot and Ivy Bridge. Clearly, measuring these counters for just the 11 GAMESS calculations considered in this paper may be an insufficient number of data points for correlation analysis. Therefore, a set of HPC application benchmarks from different scientific domains is added to the analysis. This extended set of benchmarks includes: a subset of calculations from the NAS Parallel Benchmarks (CG, FT, IS, LU and MG)[5], CoMD (molecular dynamics)[3], lulesh (shock hydrodynamics)[21], miniFE (finite element)[19], miniGhost (finite difference)[6], and smg2000 (semi-coarsening multigrid)[7]. These applications are run with different input sets to generate a total of 38 data-points to supplement the 11 data-points from GAMESS. Each data-point consists of 30 performance hardware counters that measure the number of total instructions executed, the number of floating point instructions, the number of loads/stores from L1 data cache, the number of branch instructions, the number of branches accurately predicted, etc. In the correlation analysis, all counters are first normalized for a given application by the number of total instructions executed by the application. The correlation coefficients are then calculated for each of the normalized counters, and the ratio of performance on the 8-core Moonshot to that on the 8-core Ivy Bridge. Only the counters that have absolute correlation coefficients of more than 0.6 are considered [4].

Floating Point/Integer Performance

Counters that measure floating point operations and integer operations rank among the highest: 0.7 correlation coefficient for floating point operations and 0.61 for integers. The

[4] Correlation coefficients range from -1 to +1, where -1 indicates perfect negative correlation and +1 indicates perfect positive correlation.

number of floating point operations per instruction is positively correlated to the relative performance, suggesting that floating point heavy calculations perform slower on the Moonshot, while the number of integer operations is negatively correlated to the relative performance. The floating point correlation could be attributed to 1) the faster CPU clock on the Ivy Bridge (2.6GHz versus 2.4GHz on Moonshot), and 2) different theoretical floating point operations per cycle for the two systems: 16 single precision flops per cycle (using 8 fused multiply-adds) on Ivy Bridge compared to 8 single precision flops per cycle (using 4 fused multiply-adds) on the Moonshot.

Memory Subsystem Performance

Performance counters which measure the interactions of applications with the memory subsystem also register high correlations. In particular, data load instructions are correlated to the relative performance with a coefficient of 0.6; higher data loads per instruction lead to lower relative performance on the Moonshot system. Cache and main memory read bandwidths are measured on both systems using the lmbench tool[26]. Per core main memory read bandwidth on the Ivy Bridge system is 1.45× greater than that on the Moonshot system. L1-cache read bandwidth is 2× higher on the Ivy Bridge system.

Branch Predictor Performance

Performance counters that measure branches show high correlation with relative performance, with a coefficient of 0.6. The relative performance of the Moonshot improves as more branches are accurately predicted. This suggests differences in the capabilities of the branch prediction units on the two systems. There appear to be no previous accurate and verifiable studies that look at the branch unit performance. This paper introduces a benchmark intended to measure the cost of mispredicted branches. The benchmark consists of a small loop

containing a single branch. Each path of the branch increments a counter to prevent the path from being optimized away. The direction of the branch is determined by the value of a byte read from an array, either 0 or 1. The array is indexed by the loop iteration modulo the size of the array. Each entry in the array is initialized by rand () modulo 2. The configurable parameters for the benchmark are the number of loop iterations and the size of the array.

The loop is optimized similarly on both the Ivy Bridge and Moonshot systems. The loop consists of 3 blocks of 4 instructions each. A head block reads the branch direction from the array and branches to one of the other two blocks. Each of the other two blocks increment the loop counter, increment the path counter, and branches either to the loop head or departs the loop. So, each iteration of the loop consists of 8 instructions, 2 of which are branches. One of these branches is a loop branch, which should almost always be correctly predicted, and the other is a path branch, which should have a 50% misprediction rate. Four hardware counters are collected for several configurations of the benchmark: total instructions, total cycles, branch instructions (*br_ins*) and mispredicted branches (*br_msp*). Verification of whether the benchmark causes branch mispredictions can be done by comparing *br_msp* to the number of loop iterations. If the benchmark is sufficient to break the branch predictor, then there should be approximately 1 misprediction per 2 iterations. An array of 32768 random branch directions is sufficient to obtain 0.50 branch mispredictions per iteration on the Moonshot, but only 0.46 branch mispredictions per iteration on the Ivy Bridge. An array of 65536 entries achieved 0.49 mispredictions per iteration on the Ivy Bridge. This indicates that the Ivy Bridge is more capable of predicting branch directions for this benchmark than the Moonshot.

To measure the cycle impact of each misprediction, the benchmark is modified so that every entry in the array is 0. This allows the branch predictors on both systems to nearly always predict the correct branch path. The cost of a mispredicted branch is then quantified as the difference in total cycles between the two benchmarks per *br_msp*. A branch misprediction on the Ivy Bridge increases the total cycle count by 23 cycles while a misprediction on the Moonshot increases the total cycle count by 26 cycles. As the Ivy Bridge has a pipeline of only 14-19 cycles, this also suggests that an application may experience performance loss on a branch misprediction in excess of the 14 cycles it takes to flush the pipeline.

Relationship of Findings to GAMESS Calculations

To put all of the bottleneck analysis discussion into perspective for the GAMESS calculations studied in this paper, consider two calculations that show the highest and the lowest relative performance. The research question is whether the relevant performance counters for those calculations corroborate our findings. The HF-SOSCF CIS calculation using the porphin molecule run on 8 cores is 2× slower on the Moonshot than on Ivy Bridge. The HF-SOSCF CCD calculation done using the disk based method using the silatrane molecule is 1.6× slower on the Moonshot system. In terms of the floating point operations, the porphin calculation has 461 floating point operations for every 1000 instructions, while the silatrane calculation has 390 floating point instructions for every 1000 instructions. In terms of the load operations, the porphin calculation has 274 load operations every 1000 instructions while the silatrane disk based calculation has 257. Finally, branches are predicted with 97% accuracy [5] for the porphin molecule and with 98% accuracy for the silatrane disk based calculation.

[5] Accuracy is defined here as (*br_ins*-*br_msp*)/*br_ins* x 100.

Dual Socket Ivy Bridge

Figures 9, *10*, *11* show the cross-architecture comparison results that use 16-core Ivy Bridge runs as the reference. In each Figure, the performance, energy, and EDP of executions performed across multiple core counts on the Moonshot system are normalized using the corresponding reference metric on 16-core Ivy Bridge executions. For a given calculation and core-count run on Moonshot, a value of less than 1 for performance, energy or EDP (*Figures 9*, *10*, *11*) indicates the performance (energy or EDP) on the Moonshot is better than 16-core run of the same calculation on the Ivy Bridge system. The key conclusion here is that the performance of the GAMESS calculations using 16 cores with Ivy Bridge is matched by a 32-core run on the Moonshot system, suggesting a 2:1 ratio for Moonshot core to Ivy Bridge core performance. However, as mentioned previously, Ivy Bridge dual socket executions get up to 3 times more communication bandwidth (because the communication is inter-socket) than the multi-node executions on the Moonshot (32 core runs use 4 nodes).

It was previously noted that HF-SOSCF calculations which utilize disk to store and retrieve the integrals are engaged in communication events for a significant portion of the total run time. Further analysis of the computation-communication profiles of this case reveals that the 16-core run of a silatrane disk-based calculation on the Moonshot are in communication events for 1.4× more time than the same core count run on the Ivy Bridge. Therefore, the 2:1 core-to-core Moonshot:Ivy Bridge ratio is the upper bound when it comes to the performance of GAMESS calculations considered in this paper. This ratio should improve in favor of ARMv8 with improvements in the interconnect technology.

Related Work

The potential of the ARM architecture for energy efficient HPC has been recognized by both system vendors and the academic research community. Vendors such as Cavium[8] and Hewlett-Packard[18] have started to bring new products to market with the HPC community in mind, as well as researchers reporting on their own findings of the ARM as a potential design point in HPC servers[9,34]. The Barcelona Supercomputing Center designed a testbed cluster from ARMv7 processors and a 1GbE network and stressed the need for an optimized software stack and a high performance network[31].

Performance engineers have been reporting their findings on the ARM performance on scientific codes as well[4,29,33]. Padoin et al. report the energy efficiency and performance of an ARMv7 processor on the NAS Parallel Benchmarks compared to a Sandy Bridge processor. Their findings show that while ARM uses less power, it is less energy efficient than the Sandy Bridge due to its lower performance. Laurenzano et al.[24] report the effectiveness of the ARMv7 on HPC computational benchmarks. The authors concluded that ARMv7 FP/SIMD and memory subsystem performance would need to be improved in order to be a viable option for use by the scientific community.

Multiple papers have focused on GAMESS performance on different offerings of HPC systems[13,16]. A recent study by Keipert et al.[22] compared intra-node performance and energy efficiency of GAMESS on commercially available x86, 32-bit ARM, and 64-bit ARM systems. Jundt et al.[20] adopt a machine-learning based methodology to learn the on-node architectural

bottlenecks on 64-bit ARM system. The present work is the first study that takes a comprehensive look at the inter-node performance, parallel efficiency and energy efficiency of different types of GAMESS calculations on a commercially available 64-bit ARM cluster.

Conclusions

This paper presented an analysis of the performance, parallel scalability and energy efficiency of a widely used quantum chemistry code, GAMESS, on a commercially available HP Moonshot 64-bit ARM cluster. GAMESS calculations explored in this work scale to larger core counts on the Moonshot system; the extent of the speedup is different for different types of calculations. In terms of the EDP metric, higher core count runs on Moonshot are almost always more efficient than lower count runs. These results show great promise for the co-design approach that considers using many low-power cores rather than a relatively few heavy-duty powerful cores to deliver an Exaflop system that can operate within in the 20MW power envelope.

A cross-architecture comparison of performance and energy efficiency metrics was also presented, based on the Intel Ivy Bridge system as the reference. For most of the benchmarking inputs used in the study, the performance on one node of Ivy Bridge with 16 cores is matched by a four node run (with 32 cores) on the Moonshot, notwithstanding the fact that 16-core runs on Ivy Bridge benefit from higher inter-socket communication bandwidth than the inter-node runs on the 64-bit ARM cluster. The results are interesting and encouraging given the relatively nascent entrance of ARM into the HPC world. Advancements in the compiler and the software

stacks for the 64-bit ARM architecture, which have only had a short time to evolve, will mitigate some of the performance bottlenecks in the 64-bit ARM architecture.

Acknowledgements

This work was supported in part by the Air Force Office of Scientific Research under AFOSR Award No. FA955012-1-0476 and by DoE SBIR Award No. DE-SC0013164. Sponsorship of the Department of Defense High Performance Computing Modernization Program, through a HASI grant, is gratefully acknowledged.

References

1. Server remote management with HP integrated lights out (iLO). http://tinyurl.com/o6so5bk (accessed August, 2015).
2. WattsUp? meters. https://www.wattsupmeters.com/ (accessed August, 2015).
3. CoMD proxy application. http://www.exmatex.org/comd.html (accessed August, 2015).
4. Abdurachmanov, D.; Bockelman, B.; Elmer, P.; Eulisse, R.; Knight, R.; Muzaffar, S. *Journal of Physics: Conference Series* **2015**, *601*, 1.
5. Bailey, D.H.; Barszcz, E.; Barton, J.T.; Browning, D.S.; Carter, R.L.; Dagum, L.; Fatoohi, R.A.; Frederickson, P.O.; Lasinski, T.A.; Schreiber, R.S.; Simon, H.D; Venkatakrishnan, V.; Weeratunga, S.K. *Supercompting* **1991**, 158.
6. Barett, R.F.; Vaughan, C.T.; Heroux, M.A. Sandia National Laboratories, **2012**, Tech. Rep. SAND 5294832.

7. Brown, P.N.; Falgout, R.D.; Jones, J.E. *J Sci. Comput.* **2000**, *21(5)*, 1823.
8. Cavium ThunderX ARM processors. http://tinyurl.com/mj2ayo4 (accessed August, 2015).
9. Cloutier, M.F.; Paradis, C.; Weaver, V.M. *Proceedings of the 1st International Workshop on Hardware-Software Co-Design for High Performance Computing* **2014**, 1.
10. Dunning, T.H. *J. Chem. Phys.* **1989**, *90(2)*, 1007.
11. EP Analytics, Inc. EPAX toolkit: Binary analysis for ARM. http://epaxtoolkit.com (accessed August, 2015).
12. iPerf – The network bandwidth measurement tool. https://iperf.fr/ (accessed August, 2015).
13. Fletcher, G.D.; Fedorov, D.G.; Pruitt, S.R.; Windus, T.L.; Gordon, M.S. *J. Chem. Theory Comput.* **2012**, *8(1)*, 75.
14. Fletcher, G.D.; Schmidt, M.W.; Bode, B.M.; Gordon, M.S. *Comput. Phys. Commun.* **2000**, *128*, 190.
15. Gonzalez, R.; Horowitz, M. *IEEE J. Solid-State Circuits* **1996**, *31(9)*, 1277.
16. Gordon, M.S.; Fedorov, D.G.; Pruitt, S.R.; Slipchenko, L.V. *Chem. Rev.* **2012**, *112(1)*, 632.
17. Head-Gordon, M.; Pople, J.A.; Frisch, M.J. *Chem. Phys. Lett.* **1988**, *153 (6)*, 503-506.
18. Moonshot moves HPC closer to ARM's reach. http://tinyurl.com/pvuwnpb (accessed August, 2015).
19. Heroux, M.A.; Doerfler, D.W.; Crozier, P.S.; Willenbring, J.M.; Edwards, H.C.; Williams, A.; Rajan, M.; Keiter, E.R.; Thornquist, H.K.; Numrich, R.W. **2009**, Tech. Rep. SAND 2009-5574.

20. Jundt, A.; Cauble-Chantrenne, A.; Tiwari, A.; Peraza, J.; Laurenzano, M.; Carrington, L. *International Workshop on Energy Efficient Supercomputing (E2SC), E2SC '15* **2015**, No. 6.

21. Karlin, I.; Bhatele, A.; Keasler, J.; Chamberlain, B.; Cohen, J.; DeVito, Z.; Haque, R.; Laney, D.; Luke, E.; Wang, F.; Richards, D.; Schulz, M.; Still, C. *Proceedings of the 2013 IEEE International Symposium on Parallel and Distributed Processing* **2013**, 919.

22. Keipert, K.; Mitra, G.; Sundriyal, V.; Leang, S.S.; Sosonkina, M.; Rendell, A.P.; Gordon, M.S. *J. Chem. Theory Comput.* **2015**, *11(11)*, 5055.

23. Laurenzano, M.; Tikir, M.; Carrington, L.; Snavely, A. *Performance Analysis of Systems Software (ISPASS)* **2010**, 175.

24. Laurenzano, M.; Tiwari, A.; Jundt, A.; Peraza, J.; Ward, J.; William, A.; Campbell, R.; Carrington, L. *Euro-Par 2014 Parallel Processing*. **2014**, *8632*, 124.

25. London, K.; Moore, S.; Mucci, P.; Seymour, K.; Luczak, R. *Department of Defense Users Group Conference Proceedings*. **2001**, 18.

26. McVoy, L.; Staelin, C. *Proceedings of the 1996 annual conference on USENIX Annual Technical Conference* **1996**, 23.

27. Meswani, M.; Laurenzano, M.; Carrington, L.; Snavely, A. *High Performance Computing Modernization Program Users Group Conference (HPCMP-UGC)* **2010**, 478.

28. Olson, R.; Schmidt, M.; Gordon, M.; Rendell, A. *Supercomputing* **2003**, 41.

29. Padoin, E.L.; Pilla, L.L.; Castro, M.; Boito, F.Z.; Navaux, P.O.A.; Méhaut, J.F. *IET Comput. Digit. Tec.* **2015**, *9(1)*, 27.

30. Rajovic, N.; Puzovic, N.; Vilanova, L.; Villavieja, C.; Ramirez, A. *Proceedings of the Second Workshop on Scalable Algorithms for Large-scale Systems* **2011**, 1.

31. Rajovic, N.; Rico, A.; Puzovic, N.; Adeniyi-Jones, C.; Ramirez, A. *Future Gener. Comput. Syst.* **2014**, *36,* 322.

32. Schmidt, M.W.; Baldridge, K.K.; Boatz, J.A.; Elbert, S.T.; Gordon, M.S.; Jensen, J.H.; Koseki, S.; Matsunaga, N.; Nyugen, K.A.; Su, S.; Windus, T.L.; Dupuis, M.; Montgomery, J.A. *J. Comp. Chem.* **1993**, *14(11)*, 1347.

33. Stanisic, L.; Videau, B.; Cronsioe, J.; Degomme, A.; Marangozova-Martine, V.; Legrand, A.; Mehaut, J.F. *Design, Automation & Test in Europe Conference & Exhibition* **2013**, 475.

34. Stanley-Marbell, P.; Cabezas, V. *Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW)* **2011**, 863.

35. Tikir, M.; Laurenzano, M.; Carrington, L.; Snavely, A. *Euro-Par 2009 Parallel Processing* **2009**, *5704*, 135.

36. Weigend, F.; Häaser, M.; Patzelt, H.; Ahlrichs, R. *Chem. Phys. Lett.* **1998**, *294*, 143.

37. Whaley, R.C.; Dongarra, J.J. *Supercomputing* **1998**, 1.

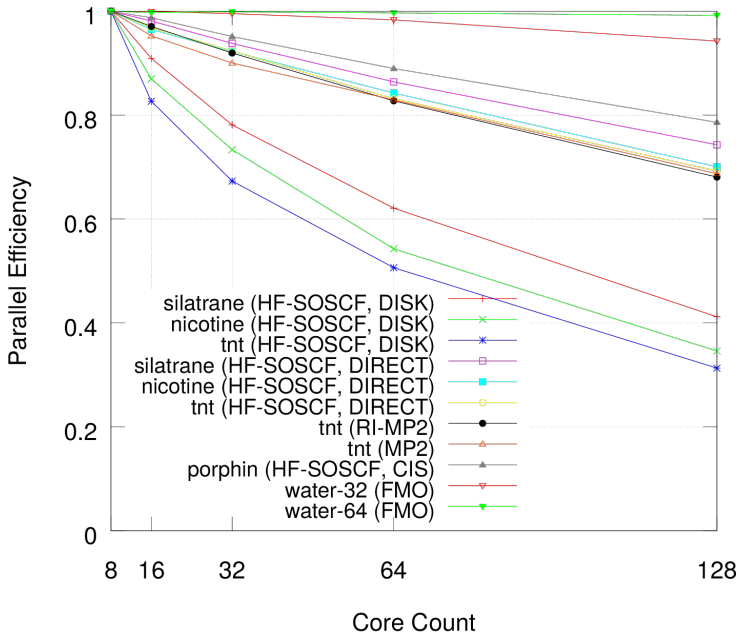


Figure 1. Moonshot results: Parallel efficiency (higher is better for large core counts).

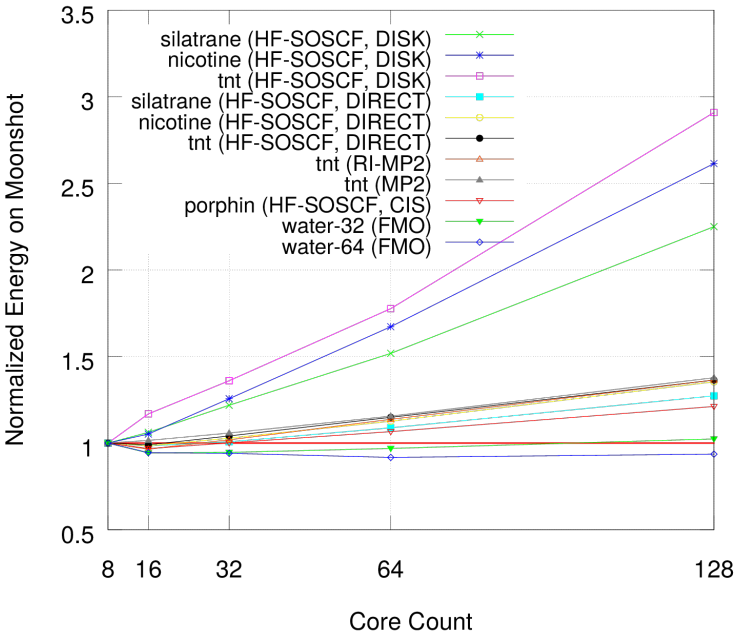


Figure 2. Moonshot results: Normalized energy consumption (lower is better for large core counts). Energy is normalized to the energy consumption for a 1-node, 8-core Moonshot run.

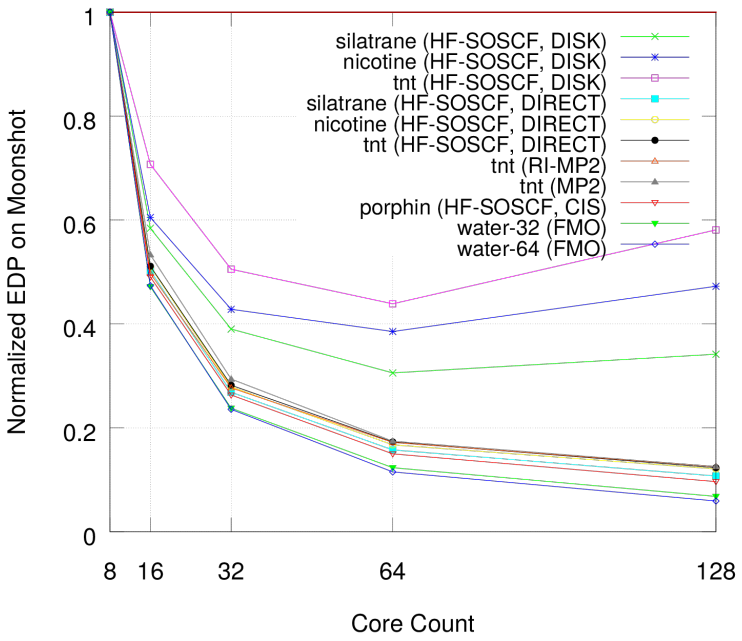


Figure 3. Moonshot results: Normalized EDP or efficiency (lower is better for large core counts). EDP is normalized to the EDP for a 1-node, 8-core Moonshot run.

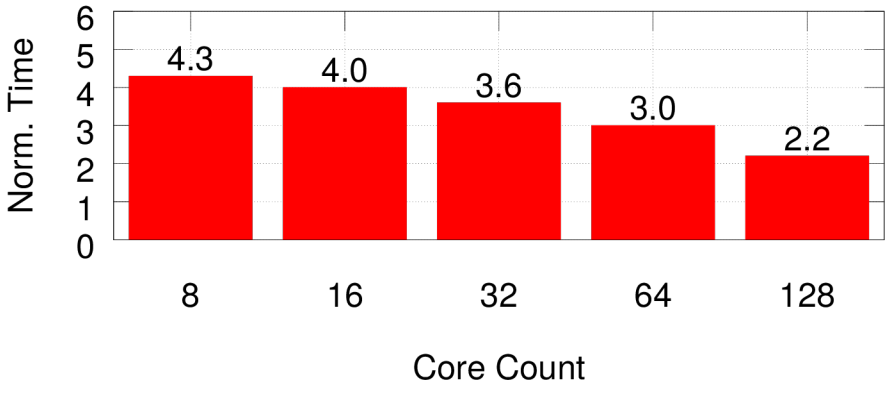
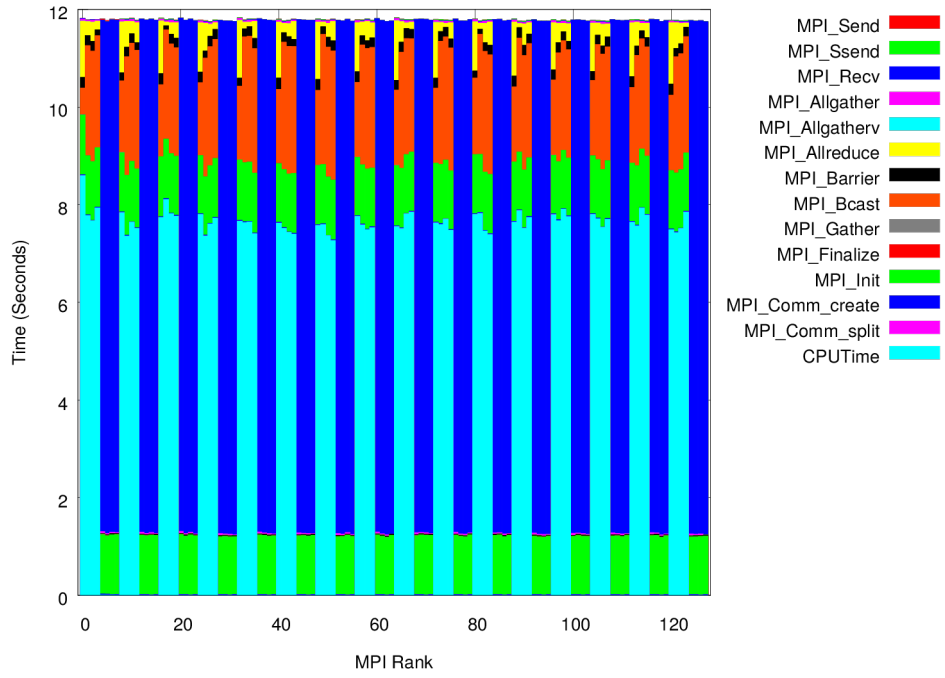


Figure 4. Moonshot results: Comparison of execution time for DISK versus DIRECT methods across multiple core counts. Execution time is normalized by the DISK method. (HF-SOSCF, silatrane).

(A)



(B)

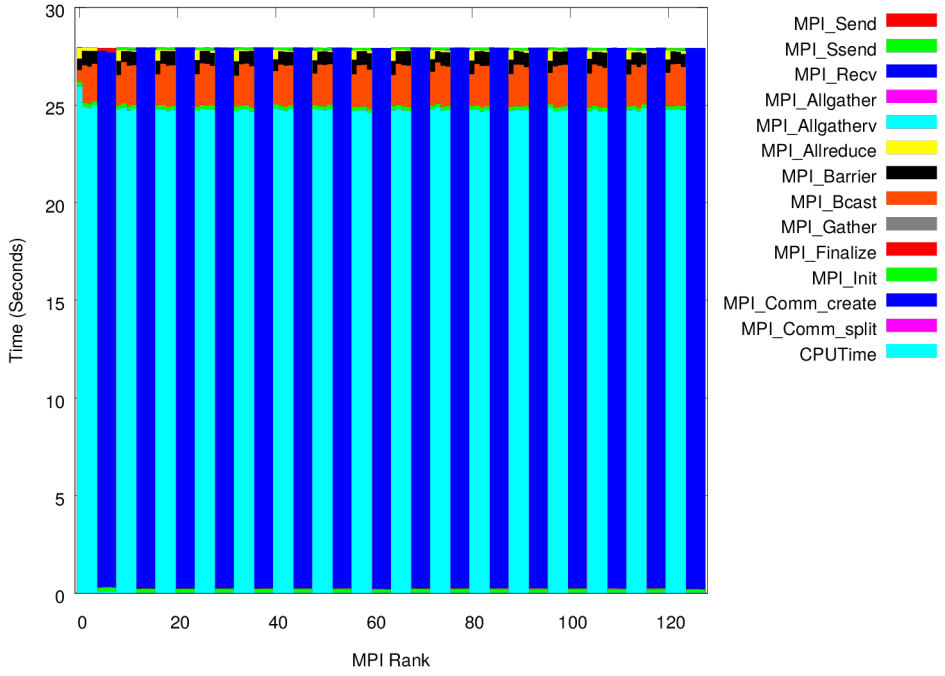


Figure 5. Moonshot results: 128-core MPI profiles for silatrane HF-SOCF energy calculation, (A) DIRECT method and (B) DISK method.

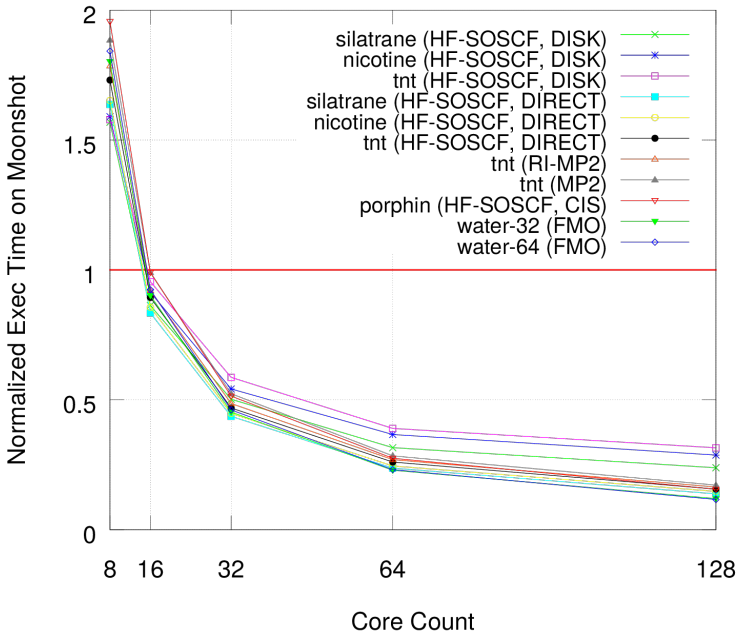


Figure 6. Cross-architectural comparison: Execution time on Moonshot normalized by execution time for 8-core, 1-socket Ivy Bridge run (lower is better for Moonshot).

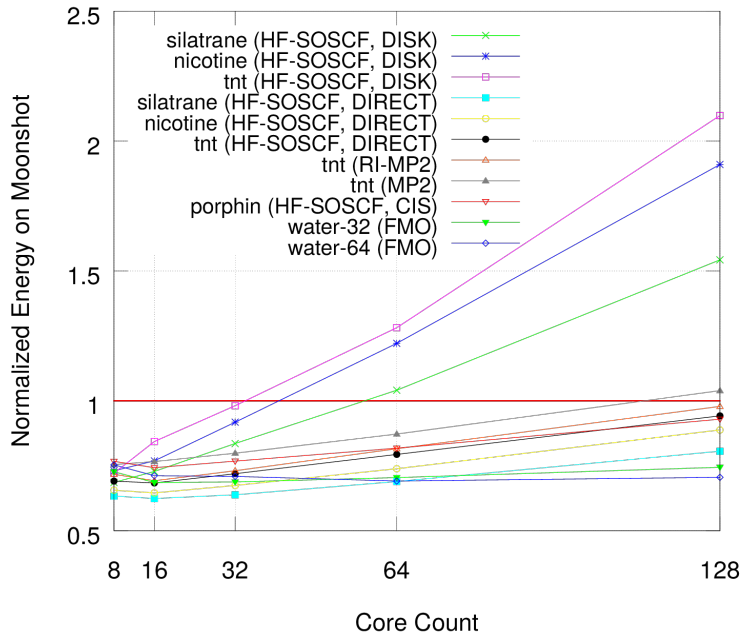


Figure 7. Cross-architectural comparison: Energy consumption on Moonshot normalized by energy consumption for 8-core, 1-socket Ivy Bridge run (lower is better for Moonshot).

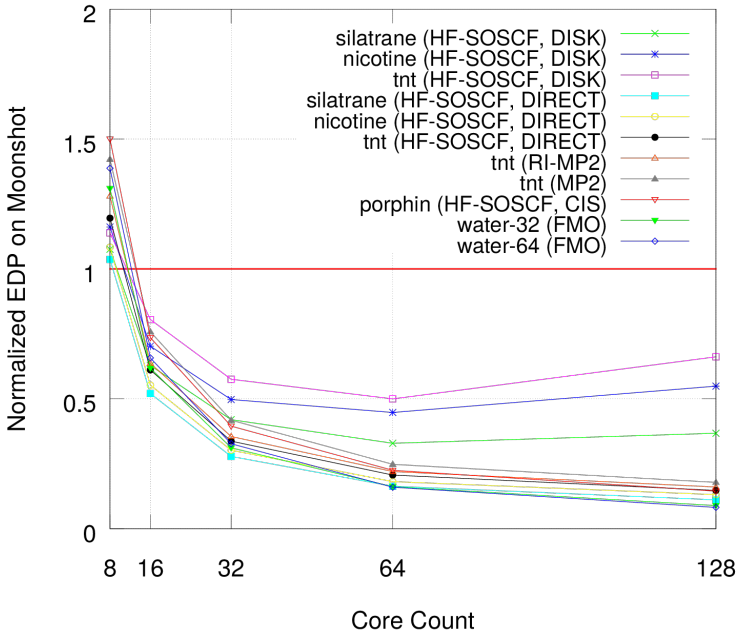


Figure 8. Cross-architectural comparison: EDP or efficiency on Moonshot normalized by EDP for 8-core, 1-socket Ivy Bridge run (lower is better for Moonshot).

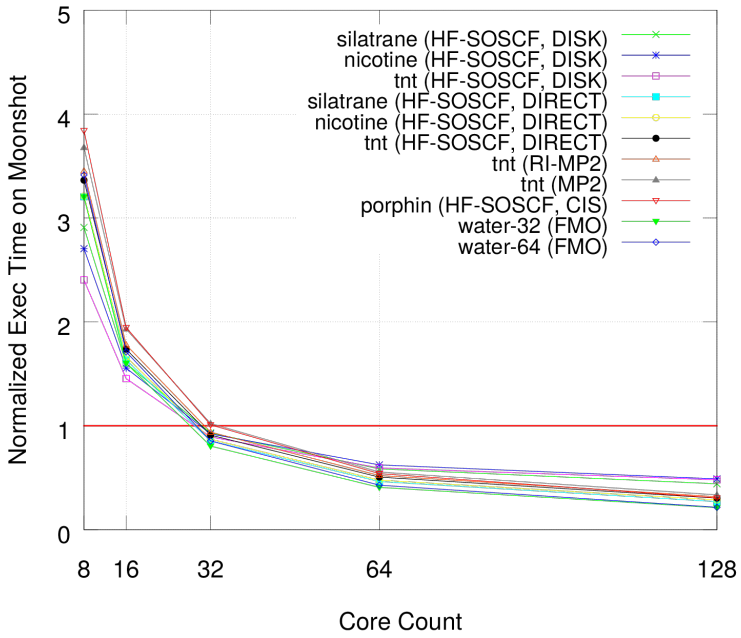


Figure 9. Execution time on Moonshot normalized by execution time for 16-core Ivy Bridge run (lower is better for Moonshot).

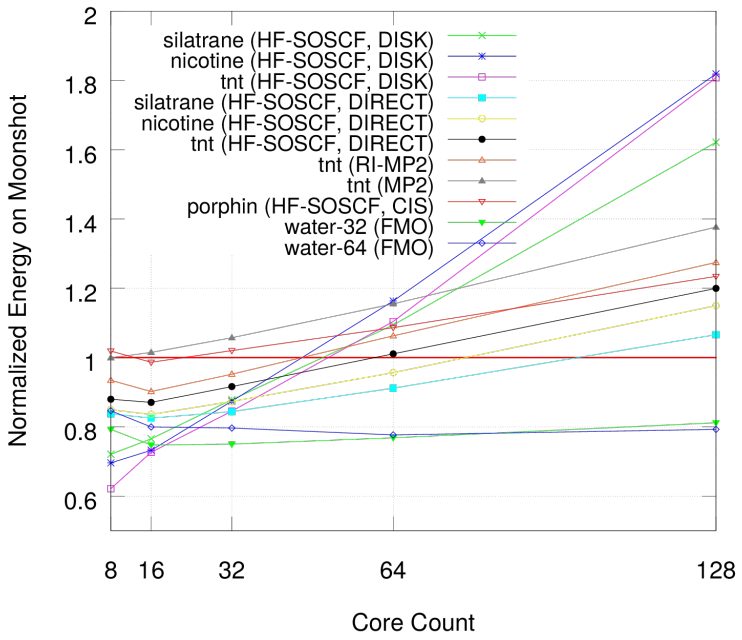


Figure 10. Energy consumption on Moonshot normalized by energy consumption for 16-core Ivy Bridge run (lower is better for Moonshot)

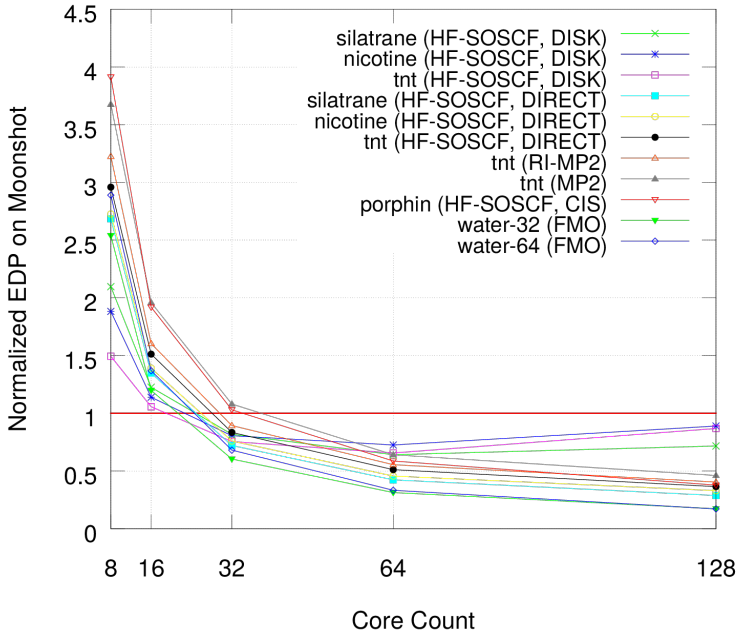


Figure 11. Cross-architectural comparison: EDP or efficiency on Moonshot normalized by EDP for 16-core Ivy Bridge run (lower is better for Moonshot)

CHAPTER 4: AN EFFICIENT MPI/OPENMP PARALLELIZATION OF THE HARTREE-FOCK METHOD FOR THE SECOND GENERATION OF INTEL XEON PHI PROCESSOR

A paper accepted for publication in the *Proceedings of the 2017 ACM/IEEE conference on Supercomputing.*

Vladimir Mironov, Yuri Alexeev, Kristopher Keipert, Michael D'mello, Alexander Moskovsky, and Mark S. Gordon

Abstract

Modern OpenMP threading techniques are used to convert the MPI-only Hartree-Fock code in the GAMESS program to a hybrid MPI/OpenMP algorithm. Two separate implementations that differ by the sharing or replication of key data structures among threads are considered, density and Fock matrices. All implementations are benchmarked on a super-computer of 3,000 Intel Xeon Phi processors. With 64 cores per processor, scaling numbers are reported on up to 192,000 cores. The hybrid MPI/OpenMP implementation reduces the memory footprint by approximately 200 times compared to the legacy code. The MPI/OpenMP code was shown to run up to six times faster than the original for a range of molecular system sizes.

Introduction

The field of computational chemistry encompasses a wide range of empirical, semi-empirical, and *ab initio* methods that are used to compute the structure and properties of molecular systems. These methods therefore have a significant impact on not only chemistry, but materials, physics, engineering and the biological sciences as well. *Ab initio* methods are rigorously derived from quantum mechanics. In principle, *ab initio* methods are more accurate than methods with empirically fitted parameters. Unfortunately, this accuracy comes at significant computational expense. For example, the time to solution for Hartree-Fock (HF) and Density Functional Theory (DFT) methods scale as approximately $O(N^3)$, where N is the number of degrees of freedom in the molecular system. The HF solution is commonly used as a starting point for more accurate *ab initio* methods, such as second order perturbation theory and coupled-cluster theory with single, double, and perturbative triple excitations. These post-HF methods scale as $O(N^5)$ and $O(N^7)$, respectively. These computational demands clearly require efficient utilization of parallel computers to treat increasingly large molecular systems with high accuracy. Modern high performance computing hardware architecture has substantially changed over the last 10 to 15 years. Nowadays, a “manycore” philosophy is common to most platforms. For example, the Intel Xeon Phi processor can have up to 72 cores. For good resource utilization, this necessitates (hybrid) MPI+X parallelism in application software.

The subject of this work is the successful adaptation of the HF method in the General Atomic and Molecular Electronic Structure System (GAMESS) quantum chemistry package to the second generation Intel Xeon Phi processor platform. GAMESS is a free quantum chemistry

software package maintained by the Gordon research group at Iowa State University[15]. GAMESS has been cited more than 10,000 times in the literature, downloaded more than 30,000 times and includes a wide array of quantum chemistry methods. The objective here is to start with the MPI-only version of GAMESS HF and systematically introduce optimizations which improve performance and reduce the memory footprint. Many existing methods in GAMESS are parallelized with MPI. OpenMP is an attractive high-level threading application program interface (API) that is scalable and portable. The OpenMP interface conveniently enables sharing of the two major objects in the HF self-consistent field (SCF) loop: the density matrix and the Fock matrix. The density and Fock data structures account for the majority of the memory footprint of each MPI process. Indeed, since these two objects are replicated across the MPI processes, memory capacity limits can easily come into play if one tries to improve the time to solution using a large number of cores. By sharing one or both of the aforementioned objects between threads, one can reduce the memory footprint and more easily leverage all of the resources (cores, fast memory etc.) of the Intel Xeon Phi processor. Reducing the memory footprint is also expected to lead to better cache utilization, and, therefore, enhanced performance.

Two hybrid OpenMP/MPI implementations of the publicly available version of the GAMESS (MPI-only) code base were constructed for this work. The first version is referred to as the “shared density private Fock”, or “private Fock” version of the code. The second version is referred to as the “shared density shared Fock”, or “shared Fock” version. In the following section, a brief survey of related work is presented. Next, key algorithmic features of the HF SCF method are discussed. Then, a description of the computer hardware test bed that was used for

benchmarking purposes is presented. An explanation of the code transformations employed in the hybrid implementation in this work follows. Next, the memory and time-to-solution results of the hybrid approach are shown. Results on up to 3,000 Intel Xeon Phi processors are presented for a range of chemical system sizes. The work ends with concluding remarks and a discussion of directions for future work.

Related Work

The HF algorithm has been a primary parallelization target since the onset of parallel computing. The primary computational components of the HF algorithm are construction of the density and Fock matrices, which are described within this work. The irregular task and data access patterns during Fock matrix construction bring significant challenges to efficient parallel distribution of the computation. The poor scaling of Fock matrix diagonalization is a major expense as well. Linear scaling methods like the fragment molecular orbital method (FMO) have been successfully applied to thousands of atoms and CPU cores[5,27], but such methods introduce additional approximations[11,12]. In any case, fragmentations methods may benefit from optimizations of the core HF algorithm as well. Early HF parallelization efforts focused on the distributed computation of the many electron repulsion integrals (ERIs) required for Fock matrix construction via MPI or other message passing libraries. The Fock and density matrices were often replicated for each rank, and load balancing algorithms were a primary optimization target. Blocking and clustering techniques were explored in depth in a landmark[14]. Contributions from that work were implemented in the quantum chemistry package NWChem[28]. In a follow-up paper[16] a node-distributed HF implementation was introduced. In this work, both the density and

Fock matrices were distributed across nodes using globally addressable array (GA). In a more recent work UPC++ library was used to achieve this goal[22]. A similar approach was used to implement distributed data parallel HF by in the GAMESS code[4,6]. This implementation utilizes the Distributed Data Interface (DDI) message passing library[13]. To further address the load balancing issues, a work stealing technique was introduced by Liu et al.[20].

A detailed study and analysis of the scalability of Fock matrix construction and density matrix construction[10], including the effects of load imbalance, was explored in a work by Chow et al.[9]. In this work, density matrix construction was achieved by density purification techniques and the resulting implementation was scaled up to 8,100 Tianhe-2 Intel Xeon Phi first generation co-processors. In fact, a number of attempts have been made to design efficient implementations of HF for accelerators[8,9,25,26,29] and other post-HF methods[7]. A major issue in this context is the management of shared data structures between cores – in particular, the density and Fock matrices. OpenMP HF implementations with a replicated Fock matrix and shared density matrix have been explored in the work of Ishimura et al.[17] and Mironov et al.[21]. The differences between these works are in the workload distribution among MPI ranks and OpenMP threads. The current work borrows some techniques from these previous works which implement HF for accelerators. The result is a hybrid MPI/OpenMP implementation that is designed to scale well on a large number of Intel Xeon Phi processors, while at the same time managing the memory footprint and maintaining compatibility with the original GAMESS codebase.

Hartree-Fock Method

The HF method is used to iteratively solve the electronic Schrödinger equation for a many-body system. The resulting electronic energy and electronic wave function can be used to compute equilibrium geometries and a variety of molecular properties. The wave function is constructed of a finite set of basis functions suitable for algebraic representation of the integro-differential HF equations. Central to HF is an effective one-electron Hamiltonian called the Fock operator which describes electron-electron interactions by mean field theory. In computational practice, the Fock operator is defined in matrix form (Fock matrix). The HF working equations are then represented by a nonlinear eigenvalue problem called the Hartree-Fock equations:

$$FC = \varepsilon SC \tag{1}$$

where ϵ is a diagonal matrix corresponding to the electronic orbital energies, F is a Fock matrix, C is matrix of molecular orbital (MO) coefficients, and S is the overlap matrix of the atomic orbital (AO) basis set. The HF equations are solved numerically by self-consistent field (SCF) iterations. The SCF iterations are preceded by computation of an initial guess density matrix and core Hamiltonian. An initial Fock matrix is constructed from terms of the core Hamiltonian and a symmetric orthogonalization matrix. Next, the Fock matrix is diagonalized to provide the MO coefficients C. These MO coefficients are used to compute an initial guess density matrix. The SCF iterations follow, in which a new Fock matrix is constructed as a function of the guess density matrix. Diagonalization of the updated Fock matrix provides a new set of MO coefficients which are used to update the density matrix. This iterative process continues until convergence is reached, which is defined by the root-mean-squared difference of consecutive densities lying below a chosen convergence threshold.

Contrary to what one might expect, the most time-consuming part of the calculation is not the solution of the Hartree-Fock equations, but rather the construction of the Fock matrix[18]. The calculation of the Fock matrix elements can be separated into one-electron and two-electron components. The computational complexity of these two parts are formally $O(N^2)$ and $O(N^4)$, respectively. In most cases of practical interest, the calculation of the two-electron contribution to the Fock matrix occupies the majority of the overall compute time.

Optimization and Parallelization of the Hartree-Fock Method

General Considerations and Design

In this section, three implementations of the HF algorithm are presented: the original MPI algorithm[24] and two new hybrid MPI/OpenMP algorithms. As mentioned earlier, the most expensive steps in HF are the computation of ERIs and the contribution of ERIs multiplied by corresponding density elements during construction of the Fock matrix. The symmetry-unique ERIs are labeled in four dimensions over *i, j, k, l* shell1 indices. The symmetry-unique quartet shell indices are traversed during Fock matrix construction. Parallelization over the four indices is complicated by the high order of permutational symmetry for shell indices. In addition, many integrals are very small in magnitude and are screened out using the Cauchy-Schwarz inequality equation. Each ERI is used to construct six elements of the Fock matrix shown in equations. (2a)–(2f) where $(i,j \mid k,l)$ corresponds to a single ERI:

$$
\begin{aligned}
&(a) \quad F_{ij} \leftarrow (i,j \mid k,l) \bullet D_{kl} \\
&(b) \quad F_{kl} \leftarrow (i,j \mid k,l) \bullet D_{ij} \\
&(c) \quad F_{ik} \leftarrow (i,j \mid k,l) \bullet D_{jl} \\
&(d) \quad F_{jl} \leftarrow (i,j \mid k,l) \bullet D_{ik} \\
&(e) \quad F_{il} \leftarrow (i,j \mid k,l) \bullet D_{jk} \\
&(f) \quad F_{jk} \leftarrow (i,j \mid k,l) \bullet D_{il}
\end{aligned}
\qquad (2)
$$

The irregular storage and access of ERIs during Fock matrix construction is a significant computational challenge. Also, the Fock matrix construction is distributed among ranks, and the final Fock matrix is summed up by a reduction. A detailed explanation of the SCF implementation in GAMESS can be found elsewhere[24].

MPI-based Hartree-Fock Algorithm

The MPI parallelization in the official release of the GAMESS code is shown in *Algorithm 1.* While this implementation has been remarkably successful, it has the disadvantage of a very high memory footprint. This is because a number of data structures (including the density matrix, the atomic orbital overlap matrix, and the one and two-electron contributions to the Fock matrix) are replicated across MPI ranks. It is a major issue for processors which have a large number of cores (like the Intel Xeon Phi). For example, running 256 MPI ranks on a single Intel Xeon Phi processor increases the memory footprint for both density and Fock matrices by a factor of 256 times. This implementation is therefore severely restricted when it comes to the size of the chemical systems that can be made to fit in memory. In a typical calculation, the number of shells (see *NShells* in *Algorithm 1*) is less than one thousand. Most often, the number

can be on the order of a few hundred shells. Thus, parallelization over a two shell indices (*Algorithm 1*) frequently results in load imbalances. The HF algorithm in GAMESS was originally designed for small- to medium-sized x86 CPU architecture clusters when load balancing is not such a significant issue. However, switching to computer systems with larger parallelism (large number of compute nodes) requires a change of approach for load balancing. Multiple solutions exist for this problem. Perhaps the simplest one is to use more shell indices to increase the iteration space and improve the load balance or introduce multilevel load balancing schemes.

Hybrid OpenMP/MPI Hartree-Fock Algorithm

In this section, the hybrid MPI/OpenMP two-electron Fock matrix code implementations of the current work are described. The main goal of this implementation is to reduce the memory footprint of the MPI-based code and to improve the load balancing by utilizing the OpenMP runtime library. Modern computational cluster nodes can have a large number of cores operating on a single random access memory. In order to efficiently utilize all of the available CPU cores, it is necessary to run many threads of execution. The major disadvantage of an MPI-only HF code is that all of the data structures are replicated across MPI processes (ranks) – since to spawn a process is the only way to use a CPU core. In practice, it is found that the memory footprint gets prohibitive rather quickly as the chemical system is scaled up. It follows from *Algorithm 1* that only the Fock matrix update incurs a potential race-condition (write dependencies) when leveraging multiple threads. Other large memory objects like the density matrix, the atomic

orbital overlap matrix, and others do not exhibit this problem, because they are read-only matrices, and as a result they can be safely shared across all threads for each MPI rank.

In a first attempt, a hybrid MPI/OpenMP Hartree-Fock code was developed with the Fock matrix replicated across threads (*Algorithm 2*). This is what is referred to as the private Fock (hybrid) version of the code. In the first loop, the master thread of each MPI rank updates the *i* index. This operation is protected by implicit and explicit barriers. OpenMP parallelization is implemented over combined j and k shell loops. Joining loops provides a much larger pool of tasks and thereby alleviates any load balancing issues that may arise. To lend credence to this idea, static and dynamic schedules of OpenMP were tested for the collapsed loop. No significant difference between the various OpenMP load balancer modes was observed. The *l* loop is the same as in the original implementation of GAMESS. The last step is the same as in the MPI-based algorithm: reduction of the Fock matrix over MPI processes. Sharing all of the large matrices except the Fock matrix saves an enormous amount of memory on the multicore systems. The observed memory footprints on the latest Xeon and Xeon Phi CPUs were reduced about 5 times. However, the ultimate goal of this work is to move all of the large data structures to shared memory.

It is not straightforward to remove Fock matrix write dependencies in the OpenMP region. As shown in *Equation 2*, up to six Fock matrix elements are updated at one time by each thread. The ERI contribution is added to the three shell column-blocks of the Fock matrix simultaneously – namely the *i*, *j*, and *k* blocks. Each block corresponds to one shell and to all basis set functions associated with this shell. The main idea of the present approach is to use

thread-private storage for each of these blocks. They are used as a buffer accumulating partial Fock matrix contribution and help to avoid write dependency. Partial Fock matrix contributions are flushed to the full matrices when the corresponding shell index changes. The access pattern of the Fock matrix by k index corresponds to only one Fock matrix element. If threads have different *k* and *l* shell indices, it would be possible to skip saving data to the *k* buffer and instead, to directly update the corresponding parts of the full Fock matrix. This condition will be satisfied if OpenMP parallelization over *k* and *l* loops is used. In this case, private storage is necessary for only the *i* and *j* blocks of the Fock matrix. In the shared Fock matrix algorithm (*Algorithm 3*) the original four loops (*Algorithm 1*) are arranged into two merged index loops. The first and second loops correspond to the combined *ij* and *kl* indices, respectively. MPI parallelization is executed over the top (*ij*) loop, while OpenMP parallelization is accomplished over the inner (*kl*) loop.

In contrast to the private Fock matrix algorithm (*Algorithm 2*), this partitioning favors computer systems with a large number of MPI ranks and is the preferred strategy because this implementation of MPI iteration space is larger and the load balance is finer. By using this partitioning, it is also possible to utilize Schwarz screening across the *i* and *j* indices. Partitioning is especially important for very large jobs with very sparse ERI tensor because it allows the user to completely skip the most costly top-loop iterations. Another difference from the private Fock matrix algorithm is that the ERI contribution is now added in three places (*Algorithm 3*, lines 25-27): to the private *i* buffer (F_{ij} , F_{ik} , F_{il}), the private *j* buffer (F_{jk} , F_{jl}), and the shared Fock matrix (F_{kl}). At the end of the joint *kl*-loop, the partial Fock matrix contribution from *i* and *j* buffers needs to be added to the full Fock matrix. It is computationally expensive for a multithreaded environment because it requires explicit thread synchronization. However, it is

possible to reduce the frequency of *i* buffer flushing. After each *kl* loop, the *i* index very likely remains the same and there will be no need for *i* buffer flushing. In the present algorithm, the old *i* index is saved after the *kl* loop (*Algorithm 3*, line 33). The flushing of the *i* buffer contribution to the Fock matrix is only performed if the *i* index were changed since the last iteration. Flushing the *j* buffer is still required after each *kl* loop (*Algorithm 3*, line 31).

A special array structure is required for flushing and reducing buffers for the *i* and *j* blocks. Buffers are organized as two-dimensional arrays. The outer dimension of these arrays corresponds to threads, and the inner dimension corresponds to the data. Using Fortran notation, data is stored in matrix columns, with each thread displayed in its own column. This (column-wise) access pattern is used when threads add an ERI contribution to the buffers (*Figure 1 (A)*). The access patter is different when it is necessary to flush a buffer into the full Fock matrix. The tree-reduction algorithm is used to sum up the contribution from different columns and add them to the full Fock matrix. In this case, the access of threads to this matrix is row-wise (*Figure 1 (B)*). Padding bytes were added to the leading dimension of the array and chunking was used on the reduction step to prevent false sharing. After the buffer is flushed into the Fock matrix, it is filled in with zeroes and is ready for the next cycle.

Methodology

Description of Hardware and Software

The benchmarks reported in this paper were performed on the Intel Xeon Phi systems provided by the Joint Laboratory for System Evaluation (JLSE) and the Theta supercomputer at the Argonne Leadership Computing Facility (ALCF)[1], which is a part of the U.S. Department of Energy (DOE) Office of Science (SC) Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program[3]. Theta is a 10-peta*f*op Cray XC40 supercomputer consisting of 3,624 Intel Xeon Phi 7230 processors. Hardware details for the JLSE and Theta system are shown in *Table 1*. The Intel Xeon Phi processor used in this paper has 64 cores each equipped with L1 cache. Each core also has two Vector Processing Units, both of which need to be used to get peak performance. This is possible because the core can execute two instructions per cycle. In practical terms, this can be achieved by using two threads per core. Pairs of cores constitute a tile. Each tile has an L2 cache symmetrically shared by the core pair. The L2 caches between tiles are connected by a two dimensional mesh. The cores themselves operate at 1.3 GHz.

Beyond the L1 and L2 cache structure, all the cores in the Intel Xeon Phi processor share 16 GBytes of MCDRAM (also known as high bandwidth memory) and 192 GBytes of DDR4. Œe bandwidth of MCDRAM is approximately 400 GBytes/sec while the bandwidth of DDR4 is approximately 100 GBytes/sec. These two levels of memory can be configured in three different ways (or modes). The modes are referred to as Flat mode, Cache mode, and Hybrid mode. Flat

mode treats the two levels of memory as separate entities. The Cache mode treats the MCDRAM as a direct mapped L3 cache to the DDR4 layer. Hybrid mode allows the user to use a fraction of MCDRM as L3 cache allocate the rest of the MCDRAM as part of the DDR4 memory. In Flat mode, one may choose to run entirely in MCDRAM or entirely in DDR4. The ”numactl” utility provides an easy mechanism to select which memory is used. It is also possible to choose the kind of memory used via the ”memkind” API, though as expected this requires changes to the source code. Beyond memory modes, the Intel Xeon Phi processor supports five cluster modes. The motivation for these modes can be understood in the following manner: to maintain cache coherency the Intel Xeon Phi processor employs a distributed tag directory (DTD). This is organized as a set of per-tile tag directories (TDs), which identify the state and the location on the chip of any cache line. For any memory address, the hardware can identify the TD responsible for that address. The most extreme case of a cache miss requires retrieving data from main memory (via a memory controller). It is therefore of interest to have the TD as close as possible to the memory controller. This leads to a concept of locality of the TD and the memory controllers. It is in the developer’s interest to maintain the locality of these messages to achieve the lowest latency and greatest bandwidth of communication with caches.

Intel Xeon Phi supports all-to-all, quadrant/hemisphere and sub-NUMA cluster SNC-4/SNC-2 modes of cache operation. For large problem sizes, different memory and clustering modes were observed to have little impact on the time to solution for the three versions of the GAMESS code. For this reason, we simply chose the mode most easily available to us. In other words, since the choice of mode made little difference in performance, our choice of quad-Cache mode was ultimately driven by convenience (this being the default choice in our particular

environment). Our comments here apply to large problem sizes, so for small problem sizes, the user will have to experiment to find the most suitable mode(s).

Description of Chemical Systems

For benchmarks, a system consisting of parallel series of graphene sheets was chosen. This system is of interest to researchers in the area of microlubricants[19]. A physical depiction of the configuration is provided in *Figure 2*. The graphene-sheet system is ideal for benchmarking, because the size of the system is easily manipulated. Various Fock matrix sizes can be targeted by adjusting the system size. In all, five configurations of the graphene sheets system were studied. The datasets for the systems studied are labeled as follows: 0.5 nm, 1.0 nm, 1.5 nm, 2.0 nm, and 5.0 nm. *Table 2* lists size characteristics of these configurations. The same 6-31G(d) basis set (per atom) was used in all calculations. For N basis functions, the density, Fock, AO overlap, one-electron Fock matrices and the matrix of MO coefficients are N×N in size. These are the main data structures of significant size. The benchmarks performed in this work process matrices which range from 660×660 to 30,240×30,240. For each of the systems studied, *Table 2* lists the memory requirements of the three versions of GAMESS HF code. Denoting NBF as the number of basis functions, the following equations describe the asymptotic (NBF $\rightarrow \infty$) memory footprint for the studied HF algorithms:

$$(a)\quad M_{MPI} = \frac{5}{2} \bullet N_{BF}^2 \bullet N_{MPI\_per\_node}$$
$$(b)\quad M_{\mathrm{Pr}F} = (2 + N_{threads} \bullet N_{BF}^2 \bullet N_{MPI\_per\_node} \qquad (3)$$
$$(c)\quad M_{ShF} = \frac{7}{2} \bullet N_{BF}^2 \bullet N_{MPI\_per\_node}$$

where M_{MPI}, M_{PrF}, M_{ShF} denote the memory footprint of MPI-only, private Fock, and shared Fock algorithms respectively; $N_{threads}$ denotes the number of threads per MPI process for the OpenMP code, and $N_{MPI\_per\_node}$ denotes the number of MPI processes per KNL node. For OpenMP runs $N_{MPI\_per\_node} = 4$, while for MPI runs the number of MPI ranks was varied from 64 to 256. If one compares columns MPI versus Pr.F and Sh.F. in *Table 2*, you will see that the private Fock code has about a 50x smaller footprint compared to the stock MPI code. For the shared Fock code, the difference is even more dramatic with a savings of about 200 times. The ideal difference is 256 times since we compare 256 MPI ranks in the stock MPI code where all data structures are replicated versus 1 MPI rank with 256 threads for the hybrid MPI/OpenMP codes. But we introduced additional replicated structures (see *Figure 1*) and many relatively small data structures are replicated also in the MPI/OpenMP codes. This explains the difference between the ideal and observed footprints. Each of the aforementioned datasets was used to benchmark three versions of the GAMESS code. The first version is the stock GAMESS MPI-only release that is freely available on the GAMESS website[2]. The second version is a hybrid MPI/OpenMP code, derived from the stock release. This version has a shared density matrix, but a thread-private Fock matrix. The third version of the code is in turn derived from the second version; it has shared density and Fock matrices. A key objective was to see how these incremental changes allow one to manage (i.e., reduce) the memory footprint of the original code while simultaneously driving higher performance.

Results and Discussion

Single Node Performance

The second generation Intel Xeon Phi processor supports four hardware threads per physical core. Generally, more threads per core can help hide latencies inherent in an application. For example, when one thread is waiting for memory, another can use the processor. The out-of-order execution engine is beneficial in this regard as well. To manipulate the placement of processes and threads, the I_MPI_DOMAIN and KMP_AFFINITY environment variables were used. We examined the performance picture when one thread per core is utilized and when four threads per core are utilized. As expected, the benefit is highest for all versions of GAMESS for two threads (or processes) per core. For three and four threads per core, some gain is observed, albeit at a diminished level. *Figure 3* shows the scaling curves with respect to the number of hardware threads utilized observed by us.

As a first test, single-node scalability was examined with respect to hardware threads of all three versions of GAMESS. For the MPI-only version of GAMESS, the number of ranks was varied from 4 to 256. For the hybrid versions of GAMESS, the number of ranks times the number of threads per rank is the number of hardware threads targeted. The larger memory requirements of the original MPI-only code restrict the computations to, at most, 128 hardware threads. In contrast, the two hybrid versions can easily utilize all 256 hardware threads available. Finally, in general terms, on cache based memory architectures, it is expected that larger memory

footprints potentially lead to more cache capacity and cache line conflict effects. These effects can lead to diminished performance, and this is yet another motivation to look at a hybrid MPI+X approach. The results of our single-node tests are plotted in *Figure 4*. It is found that using the private Fock version leads to the best time to solution for the 1.0 nm dataset, for any number of hardware threads. This version of the code is much more memory-efficient than the stock version but, because the Fock matrix data structure is private, it has a much larger memory footprint than the shared Fock version of GAMESS. Nevertheless, because the Fock matrix is private, there is less thread contention than the shared Fock version.

It was mentioned previously that shared Fock algorithm introduces additional overhead for thread synchronization. For small numbers of Intel Xeon Phi threads, this overhead is expected to be low. Therefore the shared Fock version is expected to be on par with the other versions. Eventually, as the overhead of the synchronization mechanisms begins to increase, the private Fock version of the code is found to dominate. In the end, the private Fock version outperforms stock GAMESS because of the reduced memory footprint, and outperforms the shared Fock version because of a lower synchronization overhead. Therefore, on a single node, the private Fock version gives the best time-to-solution of the three codes, but the shared Fock version strikes a better balance between memory utilization and performance. Beyond this, one must consider the choice of memory mode and cluster mode of the Intel Xeon Phi processor. It should be noted that, depending on the compute and memory access patterns of a code, the choice of memory and cluster mode can be a potentially significant performance variable.

The performance impact of different memory and cluster modes is examined for the 0.5 nm (small) and 2.0 nm (large) datasets. The results are shown in *Figure 5*. For both datasets, some variation in performance is apparent when different cluster modes and memory modes are used. The smaller dataset indicates more sensitivity to these variables than the larger dataset. Also, for both data sizes the private Fock version performs best in all cluster and memory modes tested. Also, except in the All-to-All cluster mode, the shared Fock version significantly outperforms the MPI-only stock version. In the All-to-All mode, the MPI-only version actually outperforms the shared Fock version for small datasets, and the two versions are close to parity for large datasets. In total, it is concluded that the quadrant-cache cluster memory mode is best suited to the design of the GAMESS hybrid codes.

Multi-node Performance

It is very important to note that the total number of MPI ranks for GAMESS is actually twice the number of compute ranks because of the DDI. The DDI layer was originally implemented to support one-sided communication using MPI-1. For GAMESS developers, the benefit of DDI is convenience in programming. The downside is that each MPI compute process is complemented by an MPI data server (DDI) process, which clearly results in increased memory requirements. Because data structures are replicated on a rank-by-rank basis, the impact of DDI on memory requirements is particularly unfavorable to the original version of the GAMESS code. To alleviate some of the limitations of the original implementation, an implementation of DDI based on MPI-3 was developed[23]. Indeed, by leveraging the “native” support of one-sided communication in MPI-3, the need for a DDI process alongside each MPI

rank was eliminated. For all three versions of the code benchmarked here, no DDI processes were needed.

Figure 6 shows the multi-node scalability of the MPI-only version of GAMESS versus the private Fock and the shared Fock hybrid versions. It is important to appreciate at the outset that the multinode scalability of the original MPI-only version of GAMESS is already reasonable. For example, the code scales linearly to 256 Xeon Phi nodes, and it is really the memory footprint bottleneck that limits how well all the Xeon Phi cores on any given node can be used. This pressure is reduced in the private Fock version of the code, and it is essentially eliminated in the shared Fock version. Overall, for the 2.0 nm dataset, the shared Fock code runs about six times faster than stock GAMESS on 512 Xeon Phi processors. It resulted from the better load balance of the shared Fock algorithm that uses all four shell indices – two are used in MPI and two are used in OpenMP workload distribution. The actual timings and efficiencies are listed in *Table 3*. *Figure 7* shows the behavior of the shared Fock version of GAMESS for the 5 nm dataset. It is the largest dataset we could fit in memory on Theta. Since we run on 4 MPI ranks the memory footprint is approximately 208 GB per node. This figure displays good scaling of the code up to 3,000 Xeon Phi nodes, which is equal to 192,000 cores (64 cores per node).

Conclusion

In this paper, conversion of the MPI-only GAMESS HF code to hybrid MPI-OpenMP versions is described. The resulting hybrid implementations are benchmarked to exhibit improvements in the time-to-solution and memory footprint compared to the original MPI-only

version. The code design decisions taken here were justified and implemented in a systematic way. Focus was placed on sharing the two primary (memory consuming) objects, the density and Fock matrices, in the SCF loop among the computation units. We have discussed two new HF implementations, each of which maintains full functionality of the underlying GAMESS code. In the first version, the density matrix was shared across threads, while the Fock matrix was kept private. The second version leveraged the first step, and focused entirely on making the Fock matrix a shared object. As a result, the memory footprint of the original code was lowered systematically while improving cache utilization and time-to-solution.

Clearly, we have taken only the first steps towards an efficient hybrid HF implementation in GAMESS. In future work, we plan to tune our hybrid OpenMP/MPI code more thoroughly. Our new hybrid MPI/OpenMP codes significantly outperform the official stock MPI-only code in GAMESS. Our best case implementation has about 200 times smaller memory footprint and runs up to 6 times faster than the original MPI-only version. Both our hybrid versions also have better scalability with respect to cores and nodes on single node and multi-node Intel Xeon Phi systems respectively. It is also noted that the code optimizations reported in this paper are expected to be applicable to all previous and future generations of Intel Xeon Phi processors, as well as beneficial on the Intel Xeon multicore platform. The fact that the code already scales well on a large number of second generation Intel Xeon Phi processors enables us to help bring the promise of the “many-core” philosophy to the large scientific community that has long benefited from the extensive functionality of the GAMESS code.

Like the MPI-only version, the hybrid versions of GAMESS can be deployed on systems ranging from a single desktop to large supercomputers. In addition, the hybrid codes offer enhanced configurability and parallel granularity. Finally, the lessons learned here are applicable to virtually any code that handles non-linear partial differential equations using a matrix representation. In this paper, we treat the problem of assembling a matrix in parallel subject to highly non-regular data dependencies. Indeed, a variety of methods, such as Unrestricted Hartree Fock (UHF), Generalized Valence Bond (GVB), Density Functional Theory (DFT), and Coupled Perturbed Hartree-Fock (CPHF), all have this structure. The implementation of these methods can therefore directly benefit from this work. Beyond quantum chemistry, we note, the SCF approach shares much in common with generic non-linear solvers. We therefore conclude that the strategies discussed in this work are directly applicable to computer programs encountered in other areas of science.

Acknowledgements

This research was made possible by the resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy (DOE) Office of Science User Facility supported under Contract DE-AC02- 06CH11357. We gratefully acknowledge the computing resources provided and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory. Kristopher Keipert and Prof. Mark S. Gordon acknowledge the support of a Department of Energy Exascale Computing Project grant to the Ames Laboratory. We thank the Intel® Parallel Computing Centers program for funding. The authors would like to thank the RSC Technologies staff for discussions and assistance.

References

1. Argonne National Laboratory Leadership Computing Facility. http://www.alcf.anl.gov (accessed May, 2017)
2. The General Atomic and Molecular Electronic Structure System. http://www.msg.ameslab.gov/gamess/index.html (accessed May, 2017)
3. U.S. Department of Energy INCITE Program. http://www.doeleadershipcomputing.org/ (accessed May, 2017)
4. Alexeev, Y.; Kendall, R. A.; Gordon, M. S. *Comput. Phys. Comm.* **2002**, *143* (1), 69.
5. Alexeev, Y.; Mahajan, A.; Leyffer, S.; Fletcher, G.; Fedorov, D. G. *Supercomputing* **2012**, 1.
6. Alexeev, Y.; Schmidt, M. W.; Windus, T. L.; Gordon, M. S. *J. Comp. Chem*. **2007**, *28*(10), 1685.
7. Apra, E.; Klemm, M.; Kowalski, K. *Supercomputing* **2014**, 674.
8. Asadchev, A.; Gordon, M. S.; *J. Chem. Theory Comput.* **2012**, *8(*11), 4166.
9. Chow, E.; Liu, X.; Misra, S.; Dukhan, M.; Smelyanskiy, M.; Hammond, J. R.; Du, Y.; Liao, X.; Dubey, P. *Int. J. High Perform. Comput. Appl.* **2015**, 85.
10. Chow, E.; Liu, X.; Smelyanskiy, M.; Hammond, J. R. *J. Chem. Phys.* **2015**, *142*(10)
11. Fedorov, D.G.; Kitaura, K. *J. Phys. Chem. A* **2007**, *111*(30), 6904.
12. Fedorov, D. G.; Kitaura, K.; *The Fragment Molecular Orbital Method: Practical Applications to Large Systems.* CRC Press, Boca Raton, Fl, USA
13. Fletcher, G. D.; Schmidt, M. W.; Bode, B. M.; Gordon, M. S. *Comput. Phys. Comm.* **2000**, *128*(1), 190.

14. Foster, I. T.; Tilson, J. L.; Wagner, A. L.; Shepard, R. L.; Harrison, R. J.; Kendall, R. A.; Littlefield, R. J. *J. Comp. Chem.* **1996**, *17*(1), 109.

15. Gordon, M. S.; Schmidt, M. W. *In theory and applications of computational chemistry: The first forty years*; Elsevier: Amsterdam, The Netherlands, **2005**.

16. Harrison, R. J.; Guest, M. F.; Kendall, R. A.; Bernholdt, D. E.; Wong, A. T.; Stave, M.; Anchell, J. L.; Hess, A. C.; Littlefield, R. J.; Fann, G. L. et. al. *J. Comp. Chem.* **1996**, *17*(1), 124.

17. Ishimura, K.; Kuramoto, K.; Ikuta, Y.; Hyodo, S. *J. Chem. Theory Comput.* **2010**, *6*(4), 1075.

18. Janssen, C. L.; Nielson, I. M. B. *Parallel computing in quantumchemistry.* CRC Press, Boca Raton, Fl, USA

19. Kawai, S.; Benassi, A.; Gnecco, E.; Sode, H.; Pawlak. R.; Feng, X.; Mullen, K.; Passerone, D.; Pignedoli, C.; Ruffieux, P. et. al. *Science* **2016**, *6276,* 957.

20. Liu, X.; Patel, A.; Chow, E. *In Parallel and Distributed Processing Symposium, 2014 IEEE 28th International* 902.

21. Mironov, V.; Khrenova, M.; Moskovsky, A. *High Performance Computing: 30th International Conference* **2015**, 113.

22. Ozog, D.; Kamil, A.; Zheng, Y.; Hargrove, P.; Hammond, J. R.; Malony, A.; de Jong, W.; Yelick, K. *In Parallel and Distributed Processing Symposium* **2016**, 453.

23. Pruitt, S. private communication **2016**

24. Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comp. Chem.* **1993**, *14*(11), 1347.

25. Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2008**, *4*(2), 222.

26. Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*(4), 1004.

27. Umeda, H.; Inadomi, Y.; Watanabe, T.; Yagi, T.; Ishimoto, T.; Ikegami, T.; Tadano, H.; Sakurai, T.; Nagashima, Um. *J. Comp. Chem.* **2010**, *31*(13), 2381.

28. Valiev, M.; Bylaska, E.; Govind, N.; Kowalski, K.; Straatsma, T. P.; Van Dam, H. J. J.; Wang, D.; Nieplocha, J.; Apra, E.; Windus, T. L. et. al. *Comput. Phys. Commun.* **2010**, *181*(9), 1477.

29. Wilkinson, K. A.; Sherwood, P.; Guest, M. F.; Naidoo, K. J. *J. Comp. Chem.* **2011**, *32*(10), 2313.

```
1:  for i = 1, NShells do
2:      for j = 1, i do
3:          call ddi_dlbnext(j)              ▷ MPI DLB: get new J index
4:          for k = 1, i do
5:              k==i ? l_max ← k : l_max ← j
6:              for l = 1, l_max do
    ▷ Schwartz screening:
7:                  screened ← schwartz(i, j, k, l)
8:                  if not screened then
9:                      call eri(i, j, k, l, X_ijkl)   ▷ Calculate (i, j|k, l)
    ▷ Update process-local 2e-Fock matrix:
10:                     Fock_ij,kl, ik,jl, il,jk +=
                                X_ijkl · D_kl, ij, jl, ik, jk, il
11:                 end if
12:             end for
13:         end for
14:     end for
15: end for
    ▷ 2e-Fock matrix reduction over MPI ranks:
16: call ddi_gsumf(Fock)
```

Algorithm 1. MPI parallelization of SCF in stock GAMESS

```
1:  !$omp parallel private(j, k, l, l_max, X_ijkl) shared(I)
        reduction(+ : Fock)
2:  loop
3:      !$omp master
4:      call ddi_dlbnext(i)                ▷ MPI DLB: get new I index
5:      !$omp end master
6:      !$omp barrier
7:      !$omp do collapse(2) schedule(dynamic,1)
8:      for j = 1, i do
9:          for k = 1, i do
10:             k==i ? l_max ← k : l_max ← j
11:             for l = 1, l_max do
    ▷ Schwartz screening:
12:                 screened ← schwartz(i, j, k, l)
13:                 if not screened then
14:                     call eri(i, j, k, l, X_ijkl)   ▷ Calculate (i, j|k, l)
    ▷ Update private 2e-Fock matrix:
15:                     Fock_{ij,kl,ik,jl,il,jk} +=
                                X_ijkl · D_{kl,ij,jl,ik,jk,il}
16:                 end if
17:             end for
18:         end for
19:     end for
20:     !$omp end do
21: end loop
22: !$omp end parallel
23: call ddi_gsumf(Fock)      ▷ 2e-Fock matrix reduction over MPI
```

Algorithm 2. Hybrid MPI-OpenMP SCF algorithm; Fock matrix is replicated across all threads

(Fock matrix is private)

```
1:  mxsize ← ubound(Fock)·shellSize
2:  nthreads ← omp_get_max_threads()
3:  allocate(F_I(mxsize, nthreads, F_J(mxsize, nthreads))
4:  !$omp parallel shared(F_I, F_J, Fock) &
        private(i, j, k, l, ithread)
5:  ithread ← omp_get_thread_num()
6:  loop
7:      !$omp master
8:      call ddi_dlbnext(ij) ▷ MPI DLB: get new combined IJ index
9:      !$omp end master
10:     !$omp barrier
11:     i, j ← ij                              ▷ Deduce I and J indices
12:     kl_max ← i, j                          ▷ Deduce KL-loop limit
13:     screened ← schwartz(i, j, i, j)        ▷ I and J prescreening
14:     if not screened then
15:         if i ≠ i_old then                  ▷ If i was changed flush F_I
16:             Fock(:, i)+=∑ F_I(:, 1:nthreads)
17:             !$omp barrier
18:         end if
19:         !$omp do schedule(dynamic,1)
20:         for kl = 1, kl_max do
21:             k, l ← kl                      ▷ Deduce K and L indices
22:             screened ← schwartz(i, j, k, l) ▷ Schwartz screening
23:             if not screened then
24:                 call eri(i, j, k, l, X_ijkl)   ▷ Calculate (i, j|k, l)
    ▷ Update private partial Fock matrices:
25:                 F_I(:, ithread)_j,k,l+=X_ijkl · D_kl,jl,jk
26:                 F_J(:, ithread)_k,l+=X_ijkl · D_il,ik
    ▷ Update shared Fock matrix:
27:                 Fock(k, l)+=X_ijkl · D(i, j)
28:             end if
29:         end for
30:         !$omp end do
31:         Fock(:, j)+=∑ F_J(:, 1:nthreads)          ▷ Flush F_J
32:         !$omp barrier
33:         i_old ← i
34:     end if
35: end loop
    ▷ Flush remainder F_i contribution to Fock:
36: Fock(:, i)+=∑ F_I(:, 1:nthreads)
37: !$omp end parallel
38: call ddi_gsumf(Fock)     ▷ 2e-Fock matrix reduction over MPI
```

arti

Algorithm 3. Hybrid MPI-OpenMP SCF algorithm; Fock matrix is shared across all threads

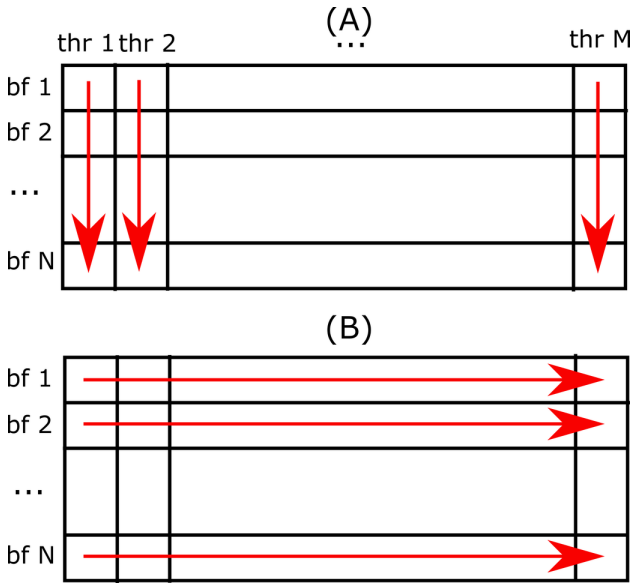


Figure 1. (A) *i* and *j* Fock vector update and (B) summation of all Fock elements from all vectors. "bf" means basis function and "thr" means thread.

Table 1. Hardware and software specifications

Intel Xeon Phi Node Characteristics	
Xeon Phi Model	7210 and 7230 (64 cores, 1.3 GHz)
Memory Per Node	16 GB MCDRAM, 192 GB DDR4
Compiler	Intel Parallel Studio XE 2016v3
JLSE Xeon Phi Cluster (26.2 TFLOPS peak)	
# Intel Xeon Phi Nodes	10
Interconnect Type	Intel Omni-Path
Theta supercomputer (9.65 PFLOPS peak)	
# Intel Xeon Phi Nodes	3,624
Interconnect Type	Aries interconnect, Dragonfly topology

Table 2. Size characteristics of chemical systems used in benchmarks

Edge Length	**# Atoms**	**# BFs**	**Memory Footprint, GB**		
			MPI	*Pr.F.*	*Sh.F.*
0.5 nm	44	660	6.5	0.1	0.5
521.0 nm	120	1800	47.6	0.9	3.8
1.5 nm	220	3300	159.6	3.1	12.9
2.0 nm	356	5340	416.8	8.2	33.7
5.0 nm	2016	30240	9869.2	208.7	1026.3

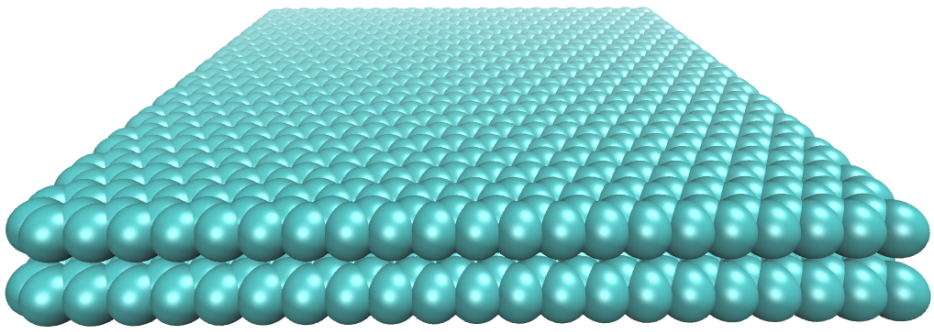


Figure 2. Model system of a C_{2016} graphene bilayer. In the text, we refer to this system as 5 nm. There are two layers with dimensions 5 nm by 5 nm. Each grapheme layer consists of 1,008 carbon atoms.

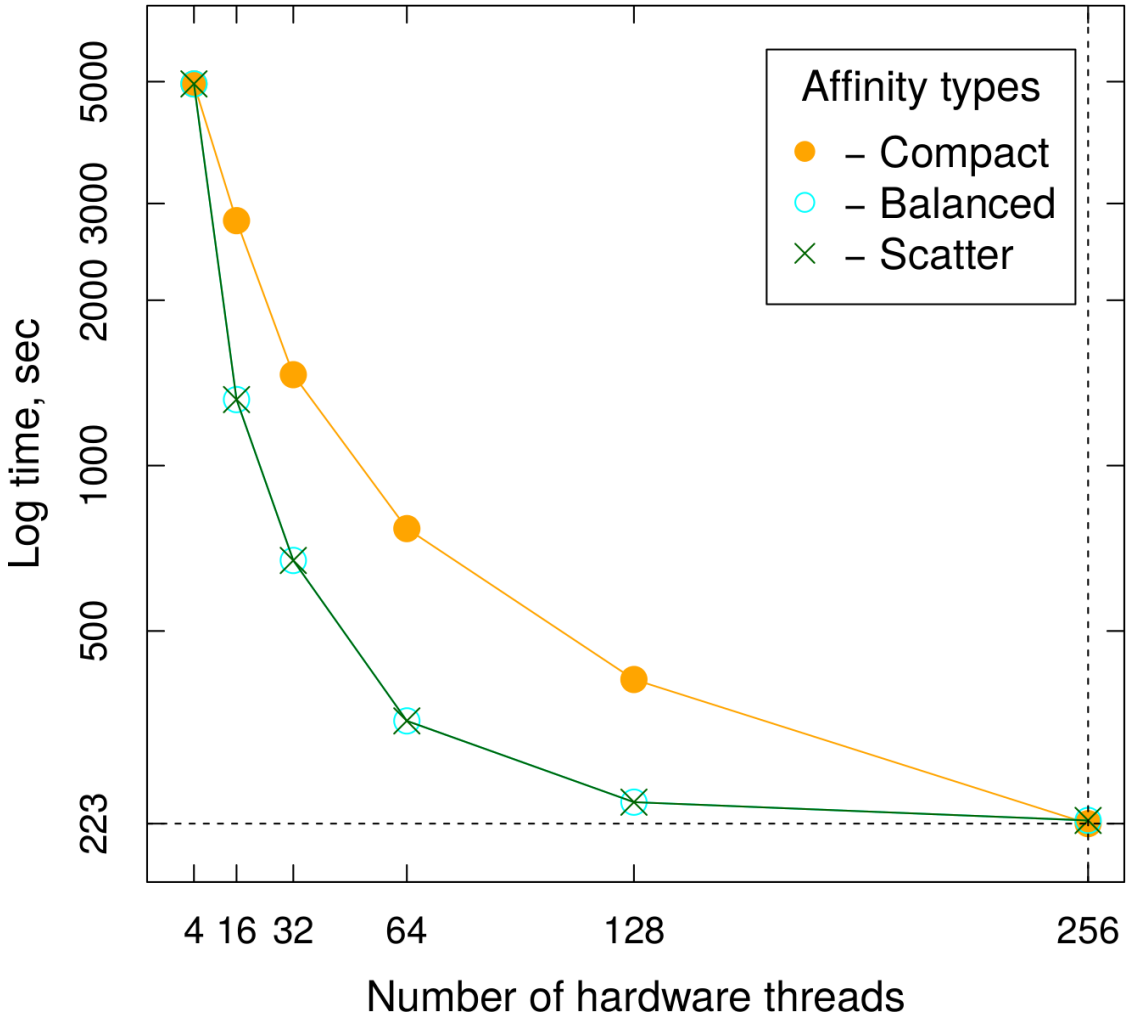


Figure 3. Performance dependence on OpenMP thread affinity type for the shared Fock version of the GAMESS code on a single Intel Xeon Phi processor, 1.0 nm benchmark. All calculations are performed in quad-cache mode. Four MPI ranks were used in all cases. The number of threads per MPI rank was varied from 1 to 64.

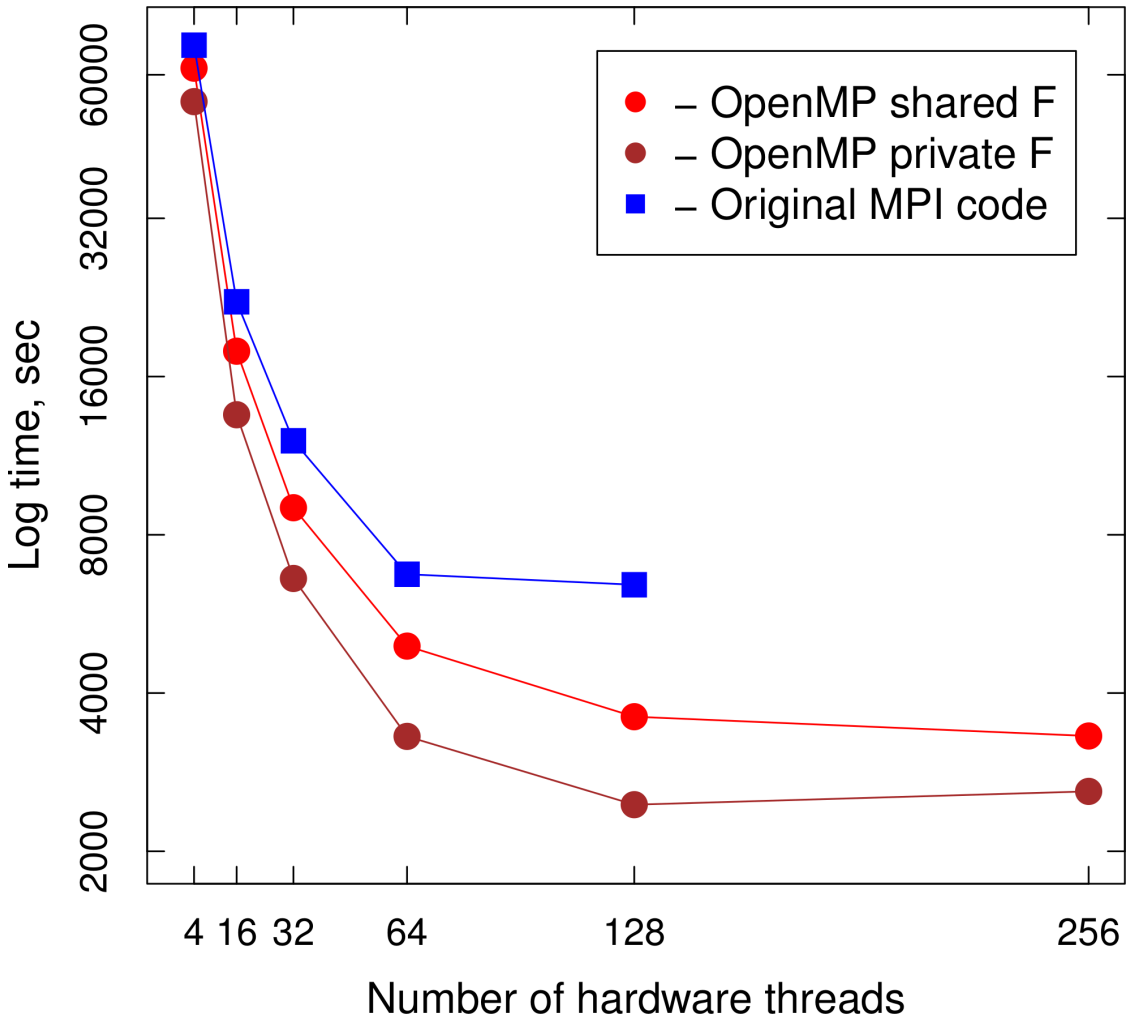


Figure 4. Scalability with respect to the number of hardware threads of the original MPI code and both MPI/OpenMP implementations on a single Intel Xeon Phi processor, 1.0 nm benchmark.

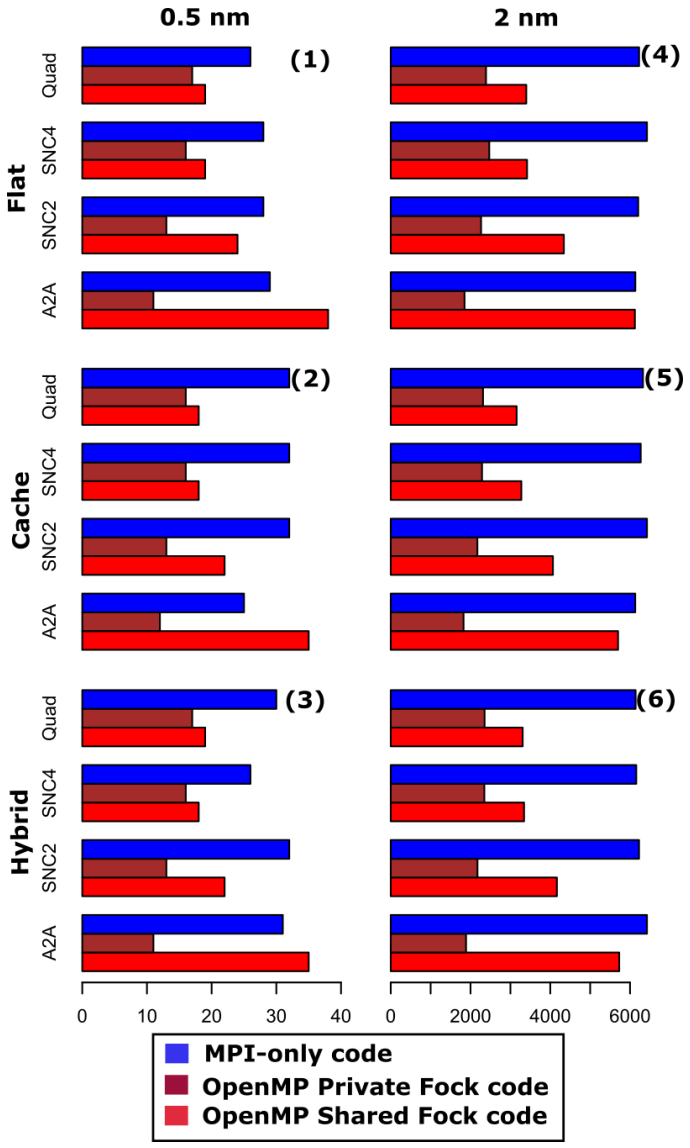


Figure 5. Time-to-solution (x-axis is time, in seconds) for different clustering and memory modes. Left column displays the smallest benchmark system (0.5 nm), and right column displays the larger 2.0 nm benchmark system.

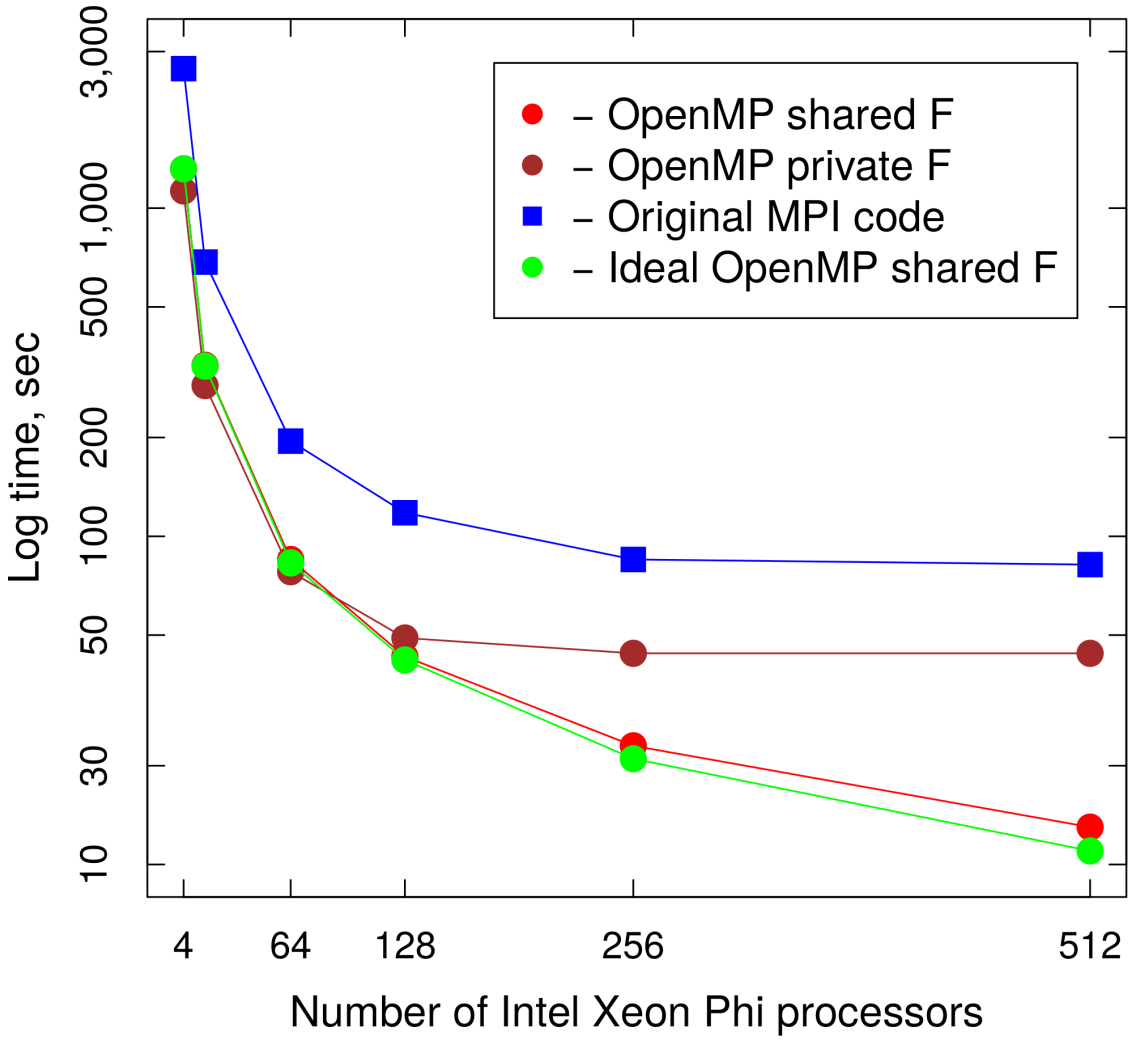


Figure 6. Multi-node scalability of the private and shared Fock implementation compared to the MPI-only GAMESS code on Theta with the 2.0 nm benchmark dataset. The quad-cache cluster-memory mode was used for all data points.

Table 3. Parallel efficiency of the three different HF algorithms, 2.0 nm benchmark dataset

# Nodes	**Time-to-solution, s**			**Parallel efficiency, %**		
	MPI	*Pr. F.*	*Sh. F.*	*MPI*	*Pr. F.*	*Sh. F.*
4	2661	1128	1318	100	100	100
16	685	288	332	97	98	99
64	195	78	85	85	90	97
128	118	49	43	70	72	96
256	85	44	23	49	40	90
512	82	44	13	25	20	79

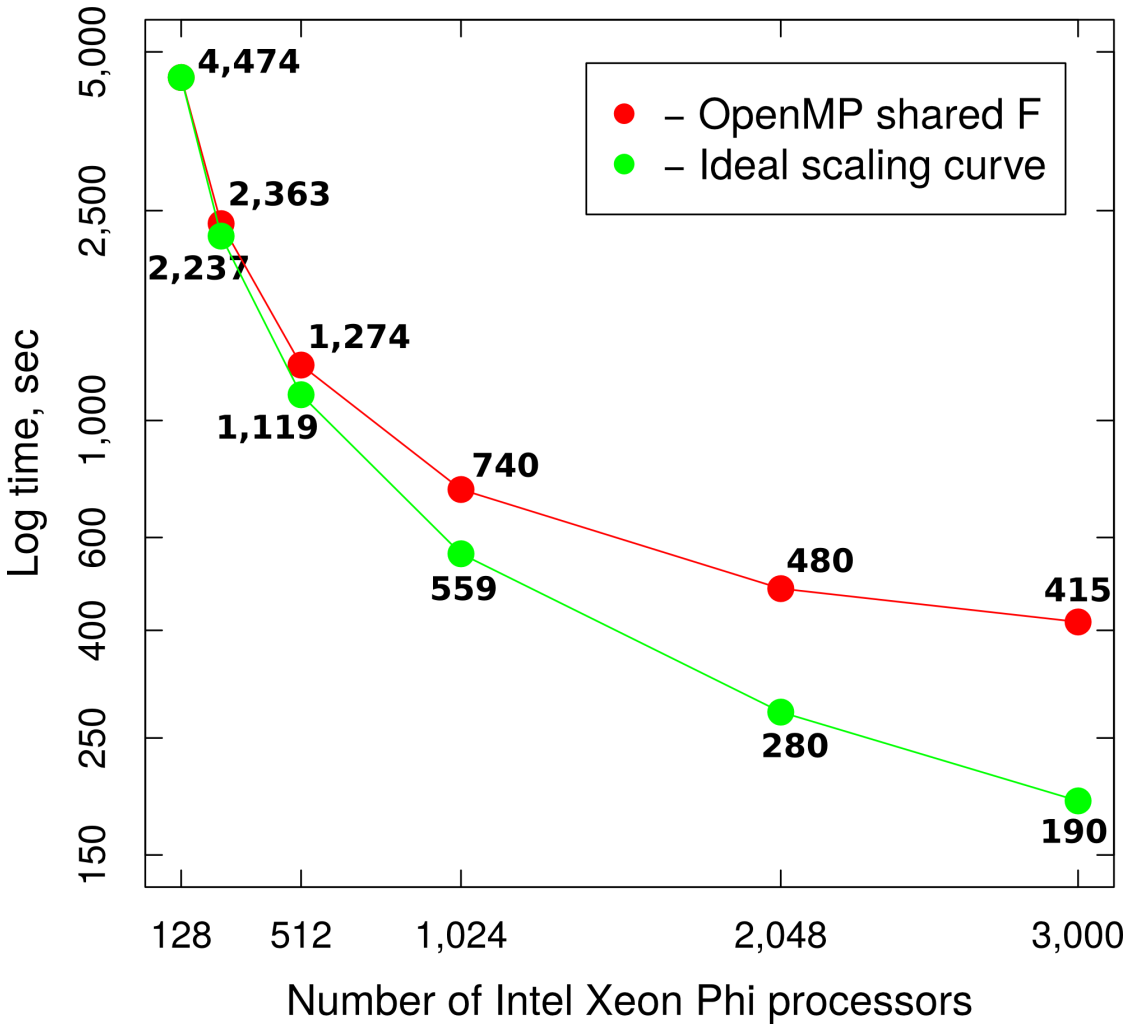


Figure 7. Scalability of the shared Fock HF implementation on Theta for the 5.0 nm benchmark dataset running on up to 3,000 Intel Xeon Phi processors. The results here are for 4 MPI ranks per node, with 64 threads per rank, giving full saturation (in terms of hardware threads) on every Intel Xeon Phi node. For each point in the figure, we show the computation time in seconds.

CHAPTER 5: INTEROPERABILITY OF ELECTRON REPULSION INTEGRAL SOLVERS WITH GAMESS

Kristopher Keipert and Mark Gordon

Abstract

The evaluation of electron repulsion integrals is a central component of many quantum chemistry computations. Isolated software libraries that contain efficient implementations of integral evaluation methods are attractive as drop-in replacements of less efficient codes. In this work, several of the key challenges of software interoperability in computational chemistry are discussed and demonstrated. Details of the integration of the ERD and SIMINT integral evaluation libraries with GAMESS are presented. Initial benchmarks of the GAMESS-ERD code show performance improvements of 6.9-14.6% for commonly used basis sets, but the computed integrals must be reordered to maintain compatibility with GAMESS. The reordering overhead essentially nullifies any observed performance benefits. A new Hartree-Fock code is written to integrate SIMINT with GAMESS, with speedups of 23.8-26.4% observed for a limited benchmark set. While the GAMESS-SIMINT interface is straightforward to implement, several valuable functionalities are lost by circumventing the GAMESS Hartree-Fock code.

Introduction

Progress in the field of computational chemistry is driven by a combination of several distinct but coupled efforts. One aspect is the development of new mathematical theories which can accurately model molecular systems and predict physical properties. Theoretical methods must be expressed as efficient computer algorithms, which requires strong familiarity with computer science and the limitations of computer hardware. The implemented computer programs are then used by trained scientists to solve various chemical problems. Through significant improvements in computational power and efficient implementations of quantum chemistry (QC) methods, accurate physical models are being applied to increasingly large chemical systems. Unfortunately, the size and complexity of QC codebases have also increased accordingly.

With a large number of QC codes in popular use, significant programming effort is expended on redundant tasks to maintain reasonable performance on new computer hardware, and to implement novel methods which already exist in other software packages. Ideally an end-user would be able to leverage the unique advantages of multiple QC codes in a single workflow. However, many existing QC applications are built on rigid software frameworks, thereby hindering easy integration. In light of these challenges, the computational QC community has recently strengthened efforts to improve software interoperability through standardized data formats and component-based software development practices. Several of those efforts will be highlighted in the present work, along with a discussion of the challenges faced when attempting

to interface a pair of two-electron integral evaluation packages with the widely used QC code GAMESS[1,2].

Software Interoperability in Quantum Chemistry

Today, dozens of molecular science software packages with unique capabilities are under active development. Many of these software packages are feature-rich and are sufficient for complete standalone computational studies. For many scientific problems, the set of methods with optimal physical accuracy and computational efficiency, as well as the desired features, may span several different programs or different physical scales of molecular systems. Furthermore, combinations of various methods may lead to new scientific discovery. In practice, even highly experienced domain software developers may face roadblocks when combining QC codes.

A major challenge is the variability of data representation in QC. Data can be broadly categorized as either metadata (MD) or large data (LD)[3-5]. MD is the human-readable information typically stored in input files (e.g. molecular geometries, basis set information, *ab initio* method parameters) and output files (e.g. electronic energy, molecular orbital coefficients). LD quantities are runtime objects often stored in binary form, with storage costs that rapidly increase with molecular system size (e.g. integrals, expansion coefficients, cluster amplitudes). While the format of LD may be somewhat constrained by computational performance considerations, a single MD quantity can be represented in dozens of ways by different scientific codes. This is clearly illustrated by the Open Babel program[6], which supports interconversion of

more than 110 chemical file formats used with various molecular science software packages. Matching up input parameters from multiple codes, whether to reproduce scientific results or support component interoperability, is often a daunting task. These same issues persist for the presentation of computed results as well. Different QC programs report different subsets of the total raw computed data, depending on what is assumed to be important to the end user, with varying formats even for equivalent results (e.g. numerical precision). These issues only grow with the rapidly increasing amount of data generated by computational chemists.

One possible approach to the incompatibility issue discussed in the previous paragraph is the development of standard formatting and nomenclature for MD. For example, molecular science domains outside of QC have developed very successful standards for specifying chemical structures such as the Protein Data Bank[7] and Crystallographic Information File[8] formats. Recently proposed standardized data formats that are tailored to QC calculations[9-11] are designed with an emphasis on efficient storage of QC information in databases. Adoption of common data standards for publication in scientific journals has been suggested as a route toward improving the reproducibility of computed results[12]. An existing approach toward MD interoperability is found in the interface between the Psi4[13]and CFOUR[14] QC codes. Psi4 includes a frontend input pre-processer that allows users to dictate flexible workflows in Python syntax. The Psi4 input may direct execution to external plugin codes linked into Psi4, or invoke other binaries directly (e.g. CFOUR). Depending on the options chosen in the Psi4 input file, a set of reasonable CFOUR parameters is chosen automatically. Advanced users can manually specify any CFOUR options in the Psi4 input file if desired. Quantities computed with CFOUR

are parsed from the output stream, and stored in Psi4 variables. This allows users to further process the results with functionalities that are available in Psi4.

Manipulation of LD is generally more challenging than MD quantities. For example, LD exists during runtime and is not usually stored after program execution. Therefore, sharing LD between QC codes requires understanding and modification of runtime workflows. Since LD is not meant to be human readable, the data formats are usually motivated by efficient storage and computational manipulation. If LD entities are shared between QC codes which require different layouts of the data, reordering the data may introduce a significant computational cost.

As with MD objects, one approach is to adopt a standard format for LD. For example, in work associated with the common component architecture (CCA) forum[15], standards were proposed for the indexing (and normalization) of a computed batch of electron repulsion integrals[16], according to a proposed standard order for Gaussian Cartesian functions. Because the layout of LD is often coupled with the implementation of the corresponding software algorithms, adopting a standardized format can require significant effort for existing codes.

In another approach, the Q5Cost format and library were developed to facilitate data exchange between QC programs with different data layouts[3-5]. An XML-based MD format was proposed, along with a binary format for LD quantities. The LD component of the Q5Cost architecture is an abstraction for QC data built upon the HDF5 data format[17]. HDF5 was designed for portable, efficient storage and I/O access of general scientific data, particularly in high performance computing environments. In order to minimize the effort required by QC

software developers, the Q5Cost library includes wrapper functions which translate data between various proprietary data formats and the Q5Cost format. This strategy is not practical for all LD quantities, as two data transformations are required when using an intermediate format, compared to a single transformation for a 1:1 interface between proprietary formats. Alternatively, QC software developers could make the effort to support the Q5Cost format natively. A standard is only valuable if it is utilized, and this "two-tier" approach can be used to add support for a common data format relatively quickly. Adoption of the Q5Cost format and library has been relatively poor, but the general approach to data interoperability is promising.

In order to facilitate integration of software components from different QC programs, the interface to use individual software components should be well defined. It is particularly difficult to isolate functionalities in QC codes such as GAMESS that were not designed using an object-oriented approach. The implementation of software interfaces also depends on the programming language used to develop the particular component. Several interfaces have been developed for a set of common QC functionalities by the CCA forum using the scientific interface description language (SIDL)[18]. The SIDL templates are used with the Babel language interoperability tool[18] to generate implementation files in the desired programming languages (Fortran, C, C++, etc.). The interface with the QC software component is then defined within the implementation files. This strategy was used to isolate the two-electron integral computation in GAMESS, with an interface computation overhead of approximately 17%[19]. In another study, a CCA-type interface was developed for the IntV3 integral package with MPQC and benchmarked for Hartree-Fock, density functional theory (DFT), and second order perturbation theory (MP2) energy and gradient calculations[16]. The interface overhead for that work ranged from 0.3-7.3%. Even

without a standardized interface, simply compartmentalizing a useful QC method can be very beneficial for software interoperability (e.g. libint[20], libxc[21], LIBEFP[22]). In this work, the integration of two standalone integral libraries with GAMESS is discussed.

Two-Electron Repulsion Integrals

The most computationally expensive component of the Hartree-Fock (HF) self-consistent field (SCF) method is the construction of the Fock matrix $F_{\mu\nu}$ by:

$$F_{\mu\nu} = H_{\mu\nu}^{core} + \sum_{\lambda\sigma}^{AO} D_{\lambda\sigma}[2(\mu\nu\,|\,\lambda\sigma) - (\mu\lambda\,|\,\nu\sigma)] \tag{1}$$

where H^{core} is the core Hamiltonian matrix, the sum runs over all atomic orbitals (AO), D is the density matrix, and μ, ν, λ, and σ are indices which range over *n* basis functions. Each (μν|λσ) quantity is an electron repulsion integral (ERI) defined by:

$$(\mu\nu\,|\,\lambda\sigma) = \int d\mathbf{r}_1 d\mathbf{r}_2 \phi_\mu(\mathbf{r}_1)\phi_\nu(\mathbf{r}_1) r_{12}^{-1} \phi_\lambda(\mathbf{r}_2)\phi_\sigma(\mathbf{r}_2) \tag{2}$$

where Φ are Gaussian basis functions centered at atomic coordinates and r_{12} is the distance $|\mathbf{r}_2-\mathbf{r}_1|$. Each Gaussian basis function is a linear combination of *k* primitive Gaussians centered on a nucleus, defined by:

$$\phi_\alpha = \sum_k N_{k\alpha} x^a y^b z^c e^{-\zeta_k r^2} \tag{3}$$

here $N_{k\alpha}$ is a contraction coefficient, x,y,z are the Cartesian coordinates of the nucleus, *a,b,c* are positive integers controlled by the angular momentum of the basis function (*L=a+b+c),* ζ is an exponent which controls the width of the orbital, and $r^2=x^2+y^2+z^2$. Basis functions are grouped into sets called shells with the same angular momentum and atomic center. For computational

efficiency, ERIs are computed at the granularity of "shell quartets" which group the basis set information of four shells. The number of ERIs computed varies for each shell quartet according to the number of basis functions contained in each shell.

The computational effort required to compute a shell quartet depends on the angular momenta of the quartet shells. An (*ff*|*ff*) quartet includes many more Gaussian functions than an (*ss*|*ss*) quartet, and each integral of the former type is generally more expensive to compute. This complicates the efficient distribution of shell quartets across computer processes. Various algorithms are available to compute ERIs such as the Obara-Saika[23,24], Rys Quadrature[25,26], and McMurchie-Davidson[27] schemes. The most computationally efficient algorithm depends on the angular momenta of the shell quartet, and the extent of Gaussian primitive contraction.

Integral Evaluation in GAMESS

The general routine for evaluation of two-electron integrals in the FORTRAN 77 GAMESS program is discussed here. A pseudocode representation of the main two-electron integral evaluation driver TWOEI is shown in *Figure 1*.

For each iteration over the inner loop, up to three symmetry-unique integral batches are computed. This is a blocking technique called "triple sort", which reduces the number and size of data transfer messages compared to canonical ordering[28]. MPI parallelization is implemented over the ISHELL and JSHELL loops, with dynamic load balancing implemented after the JSHELL loop. The innermost loop passes unscreened shell quartets to the SHELLQUART

subroutine. Depending on the angular momenta of the quartet shells, an ERI computation algorithm is chosen among the ERIC[38], rotated axis[39], and Rys Quadrature[25,26] methods in SHELLQUART. Cartesian Gaussians within shells are arranged into groups with descending powers of Cartesian products, with each group arranged in alphabetical order (e.g. X^3, Y^3, Z^3, X^2Y, X^2Z, Y^2X, Y^2Z, Z^2X, Z^2Y, XYZ). The computed integrals are immediately used to compute a partial contribution to the Fock matrix.

ERD Integral Evaluation Library

The Electron Repulsion Direct (ERD) integral library[29] is an implementation of the Rys Quadrature (RQ) method written in FORTRAN. The core idea of the RQ method is to represent the 6-dimensional ERI expression as a product of three 2-dimensional integrals that are evaluated using horizontal and vertical recurrence relations. The integrals are evaluated over an exact numerical quadrature of orthogonal Rys polynomials. The RQ method is most suitable for large angular momentum combinations because of the computational cost required to compute roots and weights of the Rys polynomials. A detailed description of the RQ method can be found elsewhere[25,26]. The ERD RQ implementation has been reported to offer improved performance over the GAMESS FORTRAN RQ code[29] with two key implementation advantages. First, the code is carefully structured to make efficient use of the CPU cache size, which is specified by the user as a tuning parameter. Second, primitive integrals are not redundantly recomputed for generally contracted basis sets. The second point is important for generally contracted basis sets, for which ERI computation has been reported to be an order of magnitude faster with ERD compared to GAMESS RQ[29].

GAMESS-ERD Integration

The GAMESS SHELLQUART routine is modified to redirect the GAMESS RQ calling function to an ERD subroutine called ERD_WRAPPER. Parameters passed to ERD_WRAPPER include the coordinates of Gaussian centers, the shell indices, a set of index parameters that assign shell information to individual shells, basis function exponents/contraction coefficients, and a buffer to store computed integrals. In comparison, the only parameters passed to the native GAMESS integral computation functions are the shell indices and output integral buffer. GAMESS is written primarily in FORTRAN 77, so large sets of parameters are shared between subroutines with common blocks. Every subroutine that uses any number of variables in the common block must include a declaration of the entire common block. Any change to a common block must be copied to every declaration of the common block throughout the code. Passing the variables as parameters to ERD_WRAPPER removes explicit dependence on the GAMESS common block within the wrapper.

Inside the ERD_WRAPPER function, some of the data passed from GAMESS must be transformed into a different format that is expected by the ERD subroutines. For example, the contraction coefficients required to compute a batch of integrals must be gathered into a single array before being passed to ERD. In GAMESS, contraction coefficients are placed in separate arrays, with each array storing all contraction coefficients corresponding to basis functions of the same angular momentum. Gaussian functions are also normalized within ERD, with no parameter exposed to the user to disable normalization. In order to avoid modification of the

ERD library, the primitive functions must be unnormalized prior to integral computation. As mentioned previously, it is usually most efficient to use multiple methods for integral computation if the basis set includes functions that range from angular momenta s and p to *d* or higher. Therefore, the GAMESS normalization routine should not simply be disabled. Instead, all contraction coefficients are copied into a static local array and unnormalized upon the first execution of ERD_WRAPPER. This strategy isolates the implementation details to the wrapper function with only a minimal memory footprint penalty. Once all required variables are stored in the appropriate format, an ERD function is called to compute the minimum amount of integer and floating point memory required to compute the integral batch. Dynamic integer and floating point arrays of the optimum size are then allocated in ERD_WRAPPER. Finally, the ERD integral evaluation function is called. Because the ordering of Cartesian functions in ERD is different from that in GAMESS, the order of computed integrals in the output buffer differs for shells with angular momentums $L>2$. For example, the Cartesian functions for *d* shells are ordered as:

$$d_{\mathrm{GAMESS}} = x^2, y^2, z^2, xy, xz, yz$$

$$d_{\mathrm{ERD}} = x^2, xy, xz, y^2, yz, z^2$$

and the integrals computed for a *dpss* shell quartet are ordered as:

$$dpss_{\mathrm{GAMESS}} = [\ (x^2\ x \mid 1\ 1), (x^2\ y \mid 1\ 1), (x^2\ z \mid 1\ 1), (y^2\ x \mid 1\ 1), (y^2\ y \mid 1\ 1), (y^2\ z \mid 1\ 1), \ldots]$$

$$dpss_{\mathrm{ERD}} = [\ (x^2\ x \mid 1\ 1), (x^2\ y \mid 1\ 1), (x^2\ z \mid 1\ 1), (xy\ x \mid 1\ 1), (xy\ y \mid 1\ 1), (xy\ z \mid 1\ 1), \ldots]$$

Modifying GAMESS or ERD to add support for a shared integral ordering is not trivial, and is outside the scope of an isolated external interface. A generalized routine was added to ERD_WRAPPER which reorders computed integrals before they are passed back to the GAMESS SCF driver. The overhead associated with reordering the integrals is significant,

particularly for the high angular momentum quartets for which the RQ method is best suited. The time between initiation of the SCF routine and the end of the first SCF iteration was benchmarked for the testosterone molecule with GAMESS and GAMESS-ERD (*Table 1*).

Even without reordering, the ERD speedup is relatively poor. Any speedup is essentially nullified by the reordering overhead, with GAMESS-ERD performing 6.9% worse than GAMESS-RYS in the worst case. The results are inconsistent with the benchmark comparisons between GAMESS and ERD for isolated ERI evaluation as reported in the original ERD implementation[29]. For example, computation of ERIs for staggered D_{3d} ethane using the cc-pvdz basis set[32] was reported to reduce the GAMESS computation time of 10.2 seconds down to 1.7 seconds for ERD. Across extended benchmarking of GAMESS-ERD vs. GAMESS-RYS, the largest speedup observed without reordering was 28.2% (PF_6^- anion with the NASA Ames ANO basis set). One deficiency in the GAMESS-ERD interface is that primitive integrals are still recomputed for contracted basis sets. To change this would require substantial modification of GAMESS. A standalone FORTRAN integral driver was implemented to illustrate the performance benefit of batching large generally contracted basis sets. All ERD function parameters were manually initialized for a single carbon atom with the ANO-RCC basis set[40], and the computed integral buffer was overwritten in subsequent function calls without any manipulation of the computed values. The ERD-type integral batching reduced the computation time from 115.7 seconds to 2.2 seconds. While this is an artificial benchmark, the performance benefit should be considered in context of the order of magnitude speedup previously reported for ERD compared to GAMESS using other generally contracted basis sets. In consideration of

the popularity of generally contracted basis sets compared to segmented contractions, efforts to implement batching in GAMESS were not pursued further.

SIMINT Integral Evaluation Library

The SIMINT integral library[34] is an implementation of the Obara-Saika[23,24] (OS) method written in the C programming language. SIMINT was written to take advantage of single-instruction, multiple-data (SIMD) vectorization capabilities of computer processors. SIMD instructions apply a single operation to multiple data points at the same time. Vectorization is becoming increasingly important as high performance computing hardware trends toward larger vector register lengths. While software compilers can automatically vectorize code in limited cases, careful manual restructuring of algorithms is usually required to maximize vectorization. The SIMINT library is built with a C++ code generator, which provides flexibility to easily modify the library. For example, the generator can be configured to optimize SIMINT code for the SIMD vector length of a target hardware system, or to change the ordering of computed integrals (including GAMESS ordering). The ability to generate complex code that is customized for hardware targets and/or software interfaces is an extremely powerful tool for performance portability and software interoperability. Several code generators have been widely adopted in computational chemistry, including other code generators for ERI evaluation[20,35].

Instead of the canonical four-index loop over shells presented *Figure 1*, the loop structure in SIMINT is implemented as a two-index loop over pairs of shells. Data corresponding to pairs of shells (e.g. coordinates of Gaussian centers, primitives and contraction coefficients) are stored

in shell pair data structures. A shell pair combines two shells corresponding to the bra or ket part of an integral quartet. Data corresponding to multiple shell pairs with the same angular momentum can be stored in a single shell pair data structure. This scheme presents two main advantages. First, several prefactors required for integral evaluation can be computed from pairs of shells in advance of the main loops over shells and primitives. Second, the integral evaluation function can operate on multiple shell quartets in one function call. SIMD registers can be efficiently utilized by filling vector lanes with primitives from different contracted shell quartets. Further details regarding the SIMINT implementation and OS method can be found elsewhere. In the present work, only the key differences between SIMINT and GAMESS that impact integration of the codes are discussed further.

GAMESS-SIMINT Integration

Supporting the loop structure over shell pairs in SIMINT requires significant modification of the GAMESS SCF driver. Because the SCF algorithm is relatively straightforward, the strategy for SIMINT-GAMESS integration is to encapsulate an entire SCF kernel in a C++ interface with GAMESS. First, the call to the GAMESS SCF driver was replaced with a conditional option to call a FORTRAN wrapper subroutine that directs execution to the GAMESS-SIMINT SCF code. The wrapper routine imports all of the input data required for the SCF routine that was initialized by GAMESS (e.g. basis set data, nuclear charges, SCF convergence tolerances) and passes the information as function parameters to the GAMESS-SIMINT SCF driver. The parameters are matched to equivalent C++ data types and specified as *const* to avoid data modifications which might impact post-Hartree-Fock routines. Next, an

initialization function is called to copy the GAMESS Gaussian shell data into simint_shell data structures. The simint_shell structures are stored in a two-dimensional C++ vector for convenience, with shells of the same angular momentum grouped into rows. GAMESS *sp* shells (pairs of s-type and p-type primitives with shared exponent values) are separated into individual *s* and *p* simint_shell structures. Next, arrays are allocated to store overlap integrals and the core Hamiltonian, and the one-electron integral driver is called. The OED one-electron integral library[29] was interfaced with GAMESS-SIMINT for this task. OED was developed concurrently with ERD, and the function arguments are almost identical to ERD. The one-electron integral driver is a four-fold loop that iterates over pairs of simint_shell vector rows and columns. Overlap, kinetic, and nuclear attraction integrals are computed for all unique pairs of shells. As with ERD, the optimum integer and floating point memory requirement are computed before each call to an integral evaluation function. For most of the OED function parameters, members of the simint_shell structs are passed directly. One exception is the basis set contraction coefficients and exponents for shell pairs, which must first be copied into a single array. Once all one-electron integrals are computed, core Hamiltonian and overlap integrals are returned to the GAMESS-SIMINT SCF driver. The typical SCF routine follows, with an initial Fock matrix formed using the core Hamiltonian as a guess, and construction of an initial density matrix.

Prior to the start of the SCF iterations, the two-electron integrals are computed with SIMINT and stored in memory. First, a two-fold loop iterates over the simint_shell vector and initializes simint_multi_shellpair (SMS) structures (corresponding to the integral bra or ket pairs). All shell pairs of a given type are grouped into the same SMS structure, but the shell pairs could potentially be distributed into separate SMS structures if desired (for example, to distribute

the work into multiple function calls during a parallel run). The SIMINT integral evaluation routine is then called within a second two-fold loop over symmetry-unique pairs of SMS structures. The output buffer of computed integrals typically contains integral values corresponding to multiple shell quartets. A final loop over the output buffer determines the *i,j,k,l* indices for each value, and stores the quantities accordingly in a one-dimensional array containing all computed integral values. Once all two-electron integrals are computed, a conventional SCF iterative procedure is executed. CBLAS/LAPACK are used for all low-level linear algebra routines throughout the GAMESS-SIMINT code, as provided by Intel MKL. The time between initiation of the SCF routine and the end of the first SCF iteration was benchmarked for the testosterone molecule with GAMESS and GAMESS-SIMINT (*Table 2*).

Modest speedups are observed with GAMESS-SIMINT for a range of basis sets. The speedups are low relative to reported speedup of the isolated SIMINT integral timings in comparison with ERD and libint[34]. This can be attributed to the overhead of the GAMESS-SIMINT interface (e.g. copying and reformatting basis set data), and possible inefficiencies in the SCF implementation compared to GAMESS. Performance profiling to identify these deficiencies is an ongoing effort. With regard to software interoperability, reimplementation of the entire Hartree-Fock routine unnecessarily reduces the functionality of GAMESS-SIMINT compared to GAMESS. For example, while individual iterations are faster with GAMESS-SIMINT, more iterations are required for convergence compared to GAMESS. GAMESS reduces the number of iterations with an improved initial orbital guess, second order orbital optimization[36], and/or extrapolation techniques such as the direct inversion of iterative subspace (DIIS)[37] method. Upon reflection, additional GAMESS functionalities could be preserved by

restricting the GAMESS-SIMINT interface to ERI evaluation and density matrix contraction only. Considering the successful performance benefits of ERI vectorization demonstrated for both SIMINT and libint, improving the performance of the GAMESS-SIMINT interface is a top priority moving forward. Adding support to the C++ interface for derivative integral evaluation by ERD is another topic of interest.

Conclusions

Software interoperability can potentially benefit the field of computational chemistry by allowing users to leverage the unique advantages of multiple QC codes in a single workflow. Differences in the layout of data structures between different QC codes can be a substantial obstacle for software integration. Various standard formats have been proposed for key QC data structures, but adding support for new data layouts in existing codes can require a substantial programming effort. A second major obstacle to interoperability is the difficulty of isolating software components from code that was not designed to be modular. This problem diminishes as QC methods are encapsulated in software libraries at an increasing rate.

In this work, software interfaces were created to integrate the ERD and SIMINT integral evaluation libraries with GAMESS. For a limited set of Hartree-Fock energy calculations for the testosterone molecule, computing integrals with ERD reduces the timing for the first SCF iteration by 6.9-14.6% for commonly used basis sets. The arrangement of integrals computed by ERD differs from GAMESS, so the integrals must be reordered. After including the performance penalty incurred by the reordering routine, the ERD speedup is essentially nullified. For the

GAMESS-SIMINT interface, an entirely new Hartree-Fock code was written. Speedups for the same benchmark set as ERD range from 19.8-26.4%. Several important GAMESS functionalities were lost by constructing a new Hartree-Fock routine, including methods that reduce the number of SCF iterations required for convergence. Writing a more minimal GAMESS-SIMINT interface with the addition of GAMESS parallelization with the distributed data interface is a focus of interest for future work.

Acknowledgements

This work was supported in part by a Department of Energy Exascale Computing Project grant. K.K. thanks Graham Fletcher and Yuri Alexeev for helpful discussions during implementation of the ERD interface. The GAMESS-ERD interface was written as a joint effort by K.K. and Alexander Findlater.

References

1. Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993**, *14*(11), 1347.
2. Gordon, M. S.; Schmidt, M. W. *In theory and applications of computational chemistry: The first forty years*; Elsevier: Amsterdam, The Netherlands, **2005**.

3. Scemama, A.; Monari, A.; Angeli, C.; Borini, S.; Evangelisti, S.; Rossi, E. *Computational Science and Its Applications – ICCSA 2008 Lecture Notes in Computer Science* **2008**, 1094–1107.
4. Borini, S.; Monari, A.; Rossi, E.; Tajti, A.; Angeli, C.; Bendazzoli, G. L.; Cimiraglia, R.; Emerson, A.; Evangelisti, S.; Maynau, D.; Sanchez-Marin, J.; Szalay, P. G. *J. Chem. Inf. Mod.* **2007**, *47*(3), 1271.
5. Angeli, C.; Bendazzoli, G. L.; Borini, S.; Cimiraglia, R.; Emerson, A.; Evangelisti, S.; Maynau, D.; Monari, A.; Rossi, E.; Sanchez-Marin, J.; Szalay, P. G.; Tajti, A. *Int. J. Quantum Chem.* **2007**, *107*(11), 2082.
6. Oboyle, N. M.; Banck, M.; James, C. A.; Morley, C.; Vandermeersch, T.; Hutchison, G. R. *J. Cheminform* **2011**, *3*(1), 33.
7. Bernstein, F. C.; Koetzle, T. F.; Williams, G. J. B.; Meyer, E. F.; Brice, M. D.; Rodgers, J. R.; Kennard, O.; Shimanouchi, T.; Tasumi, M. *FEBS J.* **1977**, *80*(2), 319.
8. Hall, S. R.; Allen, F. H.; Brown, I.D. *Acta Cryst. A.* **1991**, *47*(6), 655.
9. Hanwell, M. D.; de Jong, W. A.; Harris, C. J. *arXiv preprint:1707.04330* **2017**
10. Wang, B.; Dobosh, P. A.; Chalk, S.; Sopek, M.; Ostlund, N. S. *J. Phys. Chem. A* **2017**, *121*(1), 298.
11. Álvarez-Moreno, M.; De Graaf, C.; Lopez, N.; Maseras, N.; Poblet, J. M.; Bo, C. *J. Chem. Inf. Mod.* **2014**, *55*(1), 95.
12. Coudert, F. C. A.-X. *Chem. Mater.* **2017**, *29*(7), 2615.
13. Turney, J. M.; Simmonett, A. C.; Parrish, R. M.; Hohenstein, E. G.; Evangelista, F. A.; Fermann, J. T.; Mintz, B. J.; Burns, L. A.; Wilke, J. J.; Abrams, M. L.; Russ, N. J.; Leininger, M. L.; Janssen, C. L.; Seidl, E. T.; Allen, W. D.; Schaefer, H. F.; King, R. A.;

Valeev, E. F.; Sherrill, C. D.; Crawford, T. D. W*iley Interdisciplinary Reviews: Computational Molecular Science* **2011**, *2*(4), 556.

14. Stanton, J. F.; Gauss, J.; Harding, M. E.; Szalay, P. G.; Auer, A. A.; Bartlett, R. J.; Benedikt, U.; Berger, C.; Berndholdt, D. E.; Bomble, Y. J.; Cheng, L. **2009**
15. Kenny, J. P.; Benson, S. J.; Alexeev, Y.; Sarich, J.; Janssen, C. L.; McInnes, L. C.; Krishnan, M.; Nieplocha, J.; Jurrus, E.; Fahlstrom, C.; Windus, T. L. *J. Comput. Chem.* **2004**, *25*(14), 1717.
16. Kenny, J. P.; Janssen, C. L.; Valeev, E. F.; Windus, T. L. *J. Comput. Chem.* **2007**, *29*(4), 562.
17. Folk, M.; Heber, G.; Koziol, Q.; Pourmal, E.; Robinson, D. *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD 11* **2011**
18. Armstrong, R.; Gannon, D.; Geist, A.; Keahey, K.; Kohn, S.; Mcinnes, L.; Parker, S.; Smolinski, B. *Proceedings. The Eighth International Symposium on High Performance Distributed Computing (Cat. No.99TH8469)* **1999**
19. Peng, F.; Wu, M.-S.; Sosonkina, M.; Windus, T.; Bentz, J.; Gordon, M.; Kenny, J.; Janssen, C. *Proceedings of the 2007 symposium on Component and framework technology in high-performance and scientific computing - CompFrame 07* **2007**
20. libint. http://libint.valeyev.net/ Accessed June, 2017.
21. Marques, M. A.; Oliveira, M. J.; Burnus, T. *Comput. Phys. Commun.* **2012**, *183*(10), 2272.
22. Kaliman, I. A.; Slipchenko, L. V. *J. Comput. Chem.* **2013**, *34*(26), 2284.
23. Obara, S.; Saika, A. *J. Chem. Phys.* **1986**, *84*(7), 3963.
24. Obara, S.; Saika, A. *J. Chem. Phys.* **1988**, *89*(3), 1540.

25. Dupuis, M.; Rys, J.; King, H. F. *J. Chem. Phys.* **1976**, *65*(1), 111.

26. Rys, J.; Dupuis, M.; King, H. F. *J. Comput. Chem.* **1983**, *4*(2), 154.

27. Mcmurchie, L.; Davidson, E. *J. Comput. Phys.* **1976**, *26*(2), 218.

28. Foster, I. T.; Tilson, J. L.; Wagner, A. F.; Shepard, R. L.; Harrison, R. J.; Kendall, R. A.; Littlefield, R. J. *J. Comput. Chem.* **1996**, *17*(1), 109.

29. Flocke, N.; Lotrich, V. *J. Comput. Chem.* **2008**, *29*(16), 2722.

30. Ditchfield, R.; Hehre, W. J.; Pople, J. A. *J. Chem. Phys.* **1971**, *54*(2), 724.

31. Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56*(5), 2257.

32. Dunning, T. J. *J. Chem. Phys.* **1989**, *90*(2), 1007.

33. Krishnan, R. B.; Binkley, J. S.; Seeger, R.; Pople, J. A. *J. Chem. Phys.* **1980**, *72*(1), 650.

34. Pritchard, B. P.; Chow, E. *J. Comput. Chem.* **2016**, *37*(28), 2537.

35. Hirata, S. *J. Phys. Chem. A* **2003**, *107*(46), 9887.

36. Chaban, G.; Schmidt, M. W.; Gordon, M. S. *Theor. Chem. Acc.* **1997**, *97*(1-4), 88.

37. Pulay, P. C. A. *Chem. Phys. Lett.* **1980**, *73*(2), 393.

38. Fletcher, G. D. *Int. J. Quantum Chem.* **2006**, *106*(2), 355.

39. Ishimura, K.; Nagase, S.; *Theor. Chem. Acc.* **2008**, *120*(1-3), 186.

40. Roos, B. O.; Lindh, R.; Malmqvist, P. A.; Veryazov, V.; Widmark, P. O. *J. Phys. Chem. A* **2004**, *108*(15), 2851.

```
DO ISHELL=1, NSHELL
        DO JSHELL=1, ISHELL
                DO KSHELL=1, JSHELL
                        DO LSHELL=1, KSHELL
                                1. SCREEN INTEGRAL SHELL
                                IF (NOT SCREENED) THEN
                                        2. COMPUTE ERIs OVER UNIQUE SHELLS:
                                                (IJ|KL)
                                                (IK|JL)
                                                (IL|JK)
                                        3. UPDATE FOCK MATRIX
                                END IF
                        END DO
                END DO
        END DO
END DO
```

Figure 1. Pseudocode representation of GAMESS Integral Driver Subroutine TWOEI

Table 1. Comparison of GAMESS and GAMESS-ERD Timings for First SCF Iteration, Testosterone

Basis Set	# Basis Functions	ERD Timing	ERD-REORDER Timing	GAMESS-RYS Timing	GAMESS Timing	% Speedup
6-31G(d)[30,31]	352	73.1	81.4	78.5	50.4	6.9 / -3.7
cc-pVDZ[32]	430	321.0	350.1	375.9	211.4	14.6 / -6.9
6-311G(d,p)[33]	536	250.8	278.9	277.0	221.9	9.5 / -0.7
AUG-cc-pVDZ[32]	734	1901.4	2104.3	2148.4	1408.0	11.5 / 2.1
cc-pVTZ[32]	1090	3372.5	3571.5	3687.6	3421.2	8.6 / 3.1

All times are in seconds, running on a single Intel E5-2699 v3 core. Integral screening was disabled. ERD-REORDER includes the time required to compute and reorder all integrals to GAMESS format. For GAMESS-RYS, Rys Quadrature was used to compute all integrals. The timings in the GAMESS column are with the default GAMESS integral option, which chooses an optimum integral evaluation method based on the shell angular momentum. The speedup percentage is for (ERD / ERD-REORDER) relative to GAMESS-RYS.

Table 2. Comparison of GAMESS and GAMESS-SIMINT timings for first SCF Iteration, Testosterone

Basis Set	# Basis Functions	GAMESS-SIMINT Timing	GAMESS Timing	% Speedup
6-31G(d)[30,31]	352	38.4	50.4	23.8
cc-pVDZ[32]	430	157.6	211.4	25.4
6-311G(d,p)[33]	536	168.9	221.9	23.9
AUG-cc-pVDZ[32]	734	1129.9	1408.0	19.8
cc-pVTZ[32]	1090	2518	3421.2	26.4

All times are in seconds, running on a single Intel E5-2699 v3 core. Integral screening was disabled.

CHAPTER 6: FIRST PRINCIPLES COMPUTATIONAL INVESTIGATION OF ACETIC ACID ESTERIFICATION BY PROPYLSULFONIC ACID-FUNCTIONALIZED SILICA

A paper to be submitted to the *Journal of Physical Chemistry A*

Kristopher Keipert and Mark S. Gordon

Abstract

The reactant adsorption behavior and reaction mechanism of acetic acid esterification with methanol by acid-functionalized silica was investigated with computational methods. Reactant adsorption energies were computed for both a simple silica surface model, and an extended mesoporous silica surface. For adsorption on a single catalyst site, the computed adsorption energies for both models are in agreement. For reactant co-adsorption across two adjacent catalyst sites, it is demonstrated that the adsorption energies and adsorbate positions are sensitive to the relative orientations of the two catalyst sites. Two single-catalyst stepwise esterification reaction mechanisms were proposed, which differ primarily by the mechanism of acetic acid protonation. It is concluded that a single catalyst site is sufficient to catalyze the esterification reaction. Furthermore, the reaction mechanism following methanol pre-adsorption on the catalyst site is energetically competitive with direct protonation of acetic acid by methanol from the gas phase. Both proposed mechanisms are reversible, with similar reaction barrier heights.

Introduction

Biodiesel is a renewable alternative to diesel fuel which can be used directly in unmodified diesel engines. Almost all biodiesel is produced by base-catalyzed transesterification of fats and oils with short-chain alcohols. Over 95% of biodiesel is produced from refined edible vegetable oils, and the material cost of oil feedstocks accounts for 70-80% of total biodiesel production expenses[1]. Many cheaper alternative feedstocks such as waste cooking oils and animal fats contain a high amount of free fatty acids (FFAs). Reaction of FFAs with alkaline transesterification catalysts produces soap, which inhibits separation of the biodiesel product. Oil feedstocks with high FFA concentrations are typically pretreated with an acid-catalyzed esterification reaction which converts FFAs to methyl esters.

Acid-functionalized mesoporous silica nanoparticles (MSNs) have been identified as efficient heterogeneous catalysts for esterification of free fatty acids [2-5]. Acid-functionalized MSNs are easily separated from the reaction vessel, and may be continuously reused for multiple esterification reactions without significant loss of activity[5]. Furthermore, MSN materials exhibit high surface area, and thermal and chemical stability[6-8]. While the reaction mechanism and kinetics of homogeneous acid-catalyzed carboxylic acid esterification are well known, the reaction mechanisms proposed in the literature that utilize solid acid catalysts are inconclusive[9-13]. Two competing classes of mechanisms have been suggested, which differ primarily in the nature of the surface reaction mechanism. For the Langmuir-Hinshelwood mechanism[14,15] (LH), the carboxylic acid and alcohol reactants are both adsorbed prior to interaction. For the Eley-Rideal mechanism[16,17] (ER), only one reactant is adsorbed on the acid catalyst surface, and the

second reactant interacts directly with the adsorbed reactant. Various experimental studies have fit the reaction to either mechanism type[10,18,19]. It has been proposed that the solid-catalyst esterification mechanism type may also depend on the alcohol structure, with one work reporting an ER-type mechanism for *n*-butanol and an LH-type mechanism for ethanol[20]. In an investigation of carbinolamine formation by functionalized MSN, the MSN surface was shown to directly participate in some chemical reactions by assisting the formation of reaction intermediates[21]. In the esterification reaction of interest in this work, silica is a supporting material for the propylsulfonic acid catalyst and is not expected to participate directly in the reaction. Understanding the surface reaction mechanism serves to provide guidance for distribution of the acid catalyst over the silica support in order to maximize the catalytic surface area.

Density functional theory (DFT) has been used to computationally model adsorption behavior and chemical reactions on silica surfaces[22-25]. In a previous study[26], the single-site esterification mechanism between acetic acid and ethanol over a small silica cage functionalized with a propylsulfonic acid catalyst was investigated with DFT using the B3LYP functional[27,28]. Two mechanisms were suggested, each proceeding through two transition states in a concerted transformation. It was proposed that the acid catalyst does not protonate acetic acid as it does in the homogenous reaction mechanism, but rather acts through hydrogen bond activation of acetic acid. In the present work, DFT is used to investigate the esterification reaction of acetic acid and methanol by silica functionalized with propylsulfonic acid. Two reaction mechanisms are proposed, one each of the LH and ER type. In order to determine the importance of bulk effects when modeling adsorption on the acid-functionalized MSN surface, adsorption energies and

substrate orientations are compared between functionalized minimal silica models and extended MSN pore surface models. Additionally, reactant adsorption on single isolated catalyst sites is compared to adsorption across adjacent catalyst pairs.

Computational Details

Model Systems

The local catalytic surface of MSN is represented with both minimal model (*mm*) and embedded model (*em*) schemes. The minimal silica cluster models are shown in *Figure 1*. For a single catalytic site, the minimal model consists of a central silicon atom bonded to a propylsulfonic acid catalyst and three oxygen atoms, each terminated with SiH_3 groups. The dual-site minimal model was formed as follows. First, two single-site models were constructed from the optimized minimal model coordinates. The duplicated sites were oriented to fix the distance between the central catalyst-bonded Si atoms to 5.07Å. This distance is based on the Si-Si distance found in x-ray studies of the β-cristobalite crystal structure[29], which resembles the structure of amorphous silica. This structure has been used to model MSN materials in previous computational works[21,30]. Finally, the duplicated sites were joined by replacing a pair of adjacent terminal hydrogen atoms (one from each single-catalyst model) with an oxygen atom. For all dual-site minimal model geometry optimizations, the positions of both SiO_3 groups at the catalyst site centers were frozen. This strategy was employed to maintain the relative orientations of the propylsulfonic acid groups, thereby mimicking the surface stability of the bulk MSN surface. The structures obtained from constrained optimizations are only referenced in

discussions of reactant adsorption behavior. Geometry constraints were not used with any of the other surface models in this study.

For the second modeling scheme, the single and dual-site minimal models were embedded in a large MSN MCM-41 pore model via the surface integrated molecular orbital / molecular mechanics method (SIMOMM)[31,32] (*Figure 2*). The MCM-41 pore was previously constructed and optimized[30] with the MM3 molecular mechanics force field[33-34] and used to study benzene diffusion barriers computed with the fragment molecular orbital (FMO)[35] method. In the present study, the minimal model region is treated with quantum mechanical (QM) methods and the remaining MSN pore region is treated with molecular mechanics (MM) using the MM3 force field implemented in Tinker[36,37]. The terminal silica atoms in the QM region were capped with hydrogen atoms.

Computational Methods

All QM structures were optimized at the DFT level of theory with the M06-2X functional[38] and the 6-31G++(d,p) basis set[39,40]. All minima and transition states for fully-optimized QM structures were confirmed by calculation and diagonalization of the energy second derivatives (Hessian calculations). All references to energy values in this text are for the MP2 results, unless explicitly stated otherwise. Reactant adsorption energies (E_{ads}) were computed by the following equation:

$$E_{ads} = E_{surface/reactant(s)} - (E_{surface} + E_{reactant(s)}) \quad (1)$$

In order to focus on the atoms relevant to adsorption and esterification activities, the MSN surface and hydrogen atoms bonded to carbon are hidden in most figures which show chemical structures. All computations were performed with the GAMESS software package[41,42], and Chemcraft[43] was used for structure visualization.

Structure Notation

A simple notation is used throughout this work to refer to the various molecular structures. First, structures are prefixed with {*mm,em*}, where *mm, em* refer to minimal and embedded models. The initials *p*, *a*, *m*, and *s* are used for propylsulfonic acid, acetic acid, methanol, and silanol, respectively. The letters following *p* denote the reactants adsorbed on the propylsulfonic acid catalyst site. A numerical subscript is added to the end of the structure name if multiple structures are located with the same adsorption scheme. The structures are numbered in ascending order, from highest to lowest adsorption energy computed by MP2. As an example, consider a minimal model structure with a single acid catalyst site with methanol and acetic acid adsorbed on the surface, *mm-pam*. Four structures were located with this adsorption scheme, so the structure with the highest adsorption energy is denoted *mm-pam*$_4$. For dual-site structures, hydrogen bonding interactions are present between the catalyst sites. The hydrogen-donor catalyst is referred to by p_d, and the hydrogen acceptor site is referred to as p_a. The p_d site is listed first in the structure name. As an example, consider a minimal model dual catalyst structure located with acetic acid adsorbed on p_d, and methanol adsorbed on p_a. This structure is denoted *mm-papm*. An additional naming scheme is used to refer to propylsulfonic acid oxygen atoms. The sulfonyl hydroxy oxygen is labeled *O1*. From a top-down view of the sulfonyl group,

the oxygen clockwise from *O1* is *O2*, and the remaining oxygen is *O3*. The catalyst nomenclature scheme is illustrated in *Figure 3.*

Results and Discussion

Adsorption

Minimal Model Catalytic Surface

The adsorption energies of acetic acid and methanol were computed for single and dual-site catalyst minimal models. Optimized *mm-pa* structures are shown in *Figure 4*. Both structures demonstrate hydrogen bonding between the acetic acid carbonyl oxygen atom and sulfonyl hydroxide group, and between the acetic acid hydroxy hydrogen and a sulfonyl oxygen. The structures differ by which of the two sulfonyl oxygens is hydrogen bonded to acetic acid, with an MP2 difference in computed adsorption energy of 2.3 kcal mol^{-1}.

Optimized geometries for the *mm-pm* structures are shown in *Figure 5*. While the methanol hydroxy hydrogen may form a hydrogen bond with a sulfonyl oxygen atom (*mm-pm*$_1$), the methanol molecule is adsorbed more strongly if a hydrogen bond is formed between the methanol hydrogen and the sulfonyl hydroxy groups (e..g., (*mm-pm*$_2$)). Two structures [(*mm-pm*$_2$) and (*mm-pm*$_3$)] were located that exhibit methanol-sulfonyl hydroxyl hydrogen bonding, differing primarily by the orientation of the methanol hydroxyl hydrogen atom. For *mm-pm*$_2$, no interaction between the methanol hydroxyl hydrogen atom and the catalyst site is noted. For the *mm-pm*$_3$ structure, the analogous hydrogen atom is oriented toward *O2* with a hydrogen bond

distance of 2.04Å. The sulfur atom and all oxygen and hydrogen atoms involved in hydrogen bonding interactions for *mm-pm$_3$* form a planar ring structure.

In a previous study[10], minimal model structures resembling mm-pa$_1$ and mm-pm$_1$ were optimized with DFT/B3LYP. The adsorption energies reported are 34 kcal mol^{-1} for acetic acid and 14 kcal mol^{-1} for methanol, indicating that acetic acid is adsorbed much more strongly. The difference in adsorption strengths is in agreement with the adsorption energies for the analogous structures located in the present study (19.3 kcal mol^{-1} and 5.2 kcal mol^{-1}, respectively for DFT and 20.8 vs. 5.6 kcal mol^{-1} for MP2). However, as noted above, *mm-pm$_3$* has a much stronger MP2 binding energy of 15.2 kcal mol^{-1}. Therefore, when comparing the acetic acid complex to the most strongly bound methanol complex, the difference in adsorption energy may be as small as 3.6 kcal mol^{-1}. Given the similarity of the adsorption energies for acetic acid and methanol, reactant adsorption on free catalyst sites is expected to be competitive energetically.

Adsorption properties of functionalized mesoporous silica surfaces are affected by the presence and density of surface silanol groups[44]. Since acetic acid and methanol adsorption occurs through hydrogen bonding interactions, hydrophilic surface silanol groups may quench the adsorbates and lower the overall reaction yield. Optimized structures presented in *Figure 6* show minimal model single-site silanol structures with acetic acid and methanol hydrogen bonding interactions. The computed MP2 adsorption energies for acetic acid and methanol on silanol are nearly equal, with methanol adsorption only 1.0-1.7 kcal mol^{-1} weaker than acetic acid adsorption. One interesting observation is the presence of two hydrogen bonds between the single hydroxyl (silanol) group and acetic acid carboxyl group.

The methanol-silanol interaction in *mm-sm*$_1$ resembles the hydrogen bond in *mm-pm*$_1$, and *mm-sm*$_2$ resembles *mm-pm*$_2$. For the *mm-pm* structures, both MP2 and DFT predict that the hydrogen bond involving the methanol hydroxy hydrogen (*mm-pm*$_2$) is stronger (by approximately 7 kcal mol^{-1}) than the hydrogen bond that is formed with the methanol oxygen (*mm-pm*$_1$). For the analogous *mm-sm* structures, the adsorption energies computed with both MP2 and DFT are nearly identical. Acetic acid adsorbs significantly more strongly on the *mm-p* acid catalyst site than silanol (ΔE_{ads}=9.5 kcal mol^{-1}). For methanol, the adsorption energies for both mm-sm structures are larger than the mm-pm$_1$ adsorption energy, and smaller than the mm-pm$_2$ adsorption energy by less than 3 kcal mol^{-1}. Therefore, methanol adsorption in the vicinity of catalyst and silanol surface groups is expected to be more competitive than acetic acid (which clearly adsorbs more strongly on the catalyst). Retention of either methanol or acetic acid by silanol lowers the amount of reactant available for interaction with the acid catalyst. Ongoing experimental efforts to tune the hydrophilicity of mesoporous silica surfaces are promising routes for improving the efficiency of esterification when catalyzed by functionalized MSN.[45-47]

Optimized dual-catalyst minimal model structures and adsorption energies are shown in *Figure 7*. One *mm-ppa* structure was located, for which the acetic acid carbonyl oxygen is hydrogen bonded to the p_d hydroxy hydrogen, and the acetic acid hydroxy group is hydrogen bonded to *O2*. The adsorption energy for *mm-ppa* is a negligible 0.7 kcal mol^{-1} higher than that for the corresponding single-site structure *mm-pa*$_1$. A *mm-pap* structure in which acetic acid forms a hydrogen bond with *O3* (the more strongly bound conformation with the *mm-p* scheme) was not located. Hydrogen bonding between the catalysts in the dual-site models reduces free

rotation of the sulfonyl groups during optimization, so locating the *O2*-bound conformation is likely a result of the relative propylsulfonic acid orientation chosen for the initial optimization geometry.

One *mm-ppm* structure was located. The structure is similar to *mm-pm*$_3$, exhibiting a primary hydrogen bond between the alcohol oxygen and sulfonyl hydroxy hydrogen, and a secondary hydrogen bond between the methanol hydroxyl hydrogen and *O2*. The dual-catalyst structure also exhibits the planar orientation of the catalyst sulfur atom and hydroxy groups involved in the hydrogen bonding interactions. While the primary hydrogen bond lengths are nearly equal in the two structures, the secondary hydrogen bond length is 2.21Å for *mm-ppm* compared to 2.04Å for the single-catalyst structure. When comparing the single and dual-site minimal models, the adsorption energies for a single reactant are virtually equivalent with differences on the order of just 1 kcal mol^{-1}.

In order to describe the addition of acetic acid by methanol during the reaction of interest, it is important to know the relative orientations of the reactants when both are simultaneously adsorbed on the single (*Figure 8*) and dual-site (*Figure 9*) models. An effective reactant co-adsorption energy is computed in the same manner as the single-reactant adsorption energies, according to *Equation 1*.

The adsorption energy for *mm-pam*$_4$ is more than 7.7 kcal mol^{-1} higher than any of the other three *mm-pam* structures. The *mm-pam*$_4$ structure exhibits a hydrogen bonded ring structure between the catalyst site and both reactants. The plane of the ring is roughly perpendicular to the

plane of the silica support structure, which might be sterically hindered by an extended MSN surface. The co-adsorption energies of the remaining three *mm-pam* structures are similar, falling within a range of 1.6 kcal mol^{-1}. Structure *mm-pam₃* is the only *mm-pam* structure located that shares the dual hydrogen-bonded acetic acid conformation found for all previously discussed *mm-p/mm-pp* structures. The positions of both reactants strongly resemble those found for structure *mm-ppam.* Structures *mm-pam₁* and *mm-pam₃* are comparable, differing by the presence of a second hydrogen bond between the acetic acid hydroxy group hydrogen and *O3* for *mm-pam₃*, and by rotation of the methanoxy moiety about the methanol hydroxy bond. No optimized structure with hydrogen bonding between the *O1* oxygen and methanol was found.

Three dual-adsorbate *mm-pp* structures were located. For the *mm-ppam* system, both reactants are adsorbed onto a single catalyst site. There is no change in the inter-catalyst hydrogen bond length compared to the *mm-pp* structure without adsorbates present. The *mm-ppam* adsorbate orientations closely resemble *mm-pam₃.* In fact, the lengths of hydrogen bonds between the catalyst and adsorbates differ by less than 3% compared to *mm-pam₃*, and the co-adsorption energy is just 1.1 kcal mol^{-1} larger for *mm-ppam*. This *mm-ppam* structure demonstrates that in the presence of two catalyst sites, one catalyst may remain essentially dormant during the co-adsorption event. For the remaining two structures in *Figure 9*, reactants are adsorbed on both catalysts. The adsorption scheme for both adsorbates in *mm-papm* resemble the corresponding single-adsorbate *mm-ppa* and *mm-ppm* structures. Since the *mm-papm* p_d hydroxy hydrogen is engaged in hydrogen bonding interactions with both acetic acid (1.95Å) and p_a (2.06Å), both hydrogen bond distances are significantly longer than *mm-ppa* (1.48Å, 1.78Å, respectively).

The co-adsorption energy for the *mm-pmpa* complex is 10.9 kcal mol^{-1} greater than for *mm-papm*, and 13.8 kcal mol^{-1} greater than *mm-ppam*. In a feature unique to this structure, methanol bridges the catalysts via two hydrogen bonding interactions. The hydrogen bond distance between the methanol hydroxy hydrogen and p_a is 1.80Å, in agreement with the p_d-p_a hydrogen bond lengths found for other *mm-pp* structures. The methanol-p_d hydrogen bond distance is shorter (1.46Å) than the methanol-p_a hydrogen bond, while the catalyst S-S distance is elongated by 0.15Å compared to the average of all other *mm-pp* structures. Acetic acid adsorption on catalyst sites which participate in the p_d-p_a hydrogen bond weaken the inter-catalyst interaction, which presumably reduces the subsequent adsorption energy of methanol in the bridging orientation. The *mm-pmpa* structure is the only *mm-pp* complex that exhibits explicit interaction of one adsorbate with both catalyst sites. In summary, co-adsorption of acetic acid and methanol on separate neighboring sites is found to be energetically favorable to co-adsorption on a single site by 3 kcal mol^{-1}. With the exception of *mm-pmpa,* the *mm-p* models were sufficient for modeling adsorption behavior of individual adsorbates in terms of adsorption/co-adsorption energies and adsorbate orientations. Notably, the co-adsorption energy of *mm-pmpa* is significantly larger than that of any other *mm* structure located, and the structure cannot be modeled with a single-catalyst model.

Embedded Model Catalytic Surface

Optimized *em-p* structures with adsorbed reactants are shown in *Figure 10*. The orientation of acetic acid in the *em-pa* structure closely resembles that of the single-adsorbate acetic acid minimal model structures. The adsorption energy for the *em* structure is 17.4 kcal

mol^{-1}, compared to 17.9 kcal mol^{-1} for the analogous *mm-pa*$_1$ structure. Likewise, the geometry of the adsorbed methanol structure located with the *em* model is very similar to *mm-pm*$_3$, while the adsorption energy is lower by 2.1 kcal mol^{-1} than the *mm* structure. Stronger hydrogen bonding interactions in the *mm* structure contribute to the higher adsorption energy, as indicated by shorter distances between the alcohol oxygen and sulfonyl hydrogen (1.56Å vs. 1.59Å), and between the alcohol hydroxy hydrogen and sulfonyl carbonyl oxygen (2.04Å vs. 2.38Å). Structure *em-pam*$_1$ resembles *mm-pam*$_1$, as the co-adsorbed reactants are well-separated and just a single hydrogen bond is exhibited between acetic acid and the catalyst sulfonyl group. The adsorbate orientations for *em-pam*$_2$ most closely resemble *mm-pam*$_3$. The *em-pam*$_2$ co-adsorption energy of 24.6 kcal mol^{-1} is nearly identical to that of *mm-pam*$_3$ (24.7 kcal mol^{-1}), and the orientation of acetic acid is nearly the same for both structures. The structures differ primarily by the orientation of the methanol molecule. For the *mm* structure, the methanol hydroxy hydrogen forms a hydrogen bond with *O2* (2.05Å), and the methanol methyl group is oriented toward the acetic acid molecule. For the *em* structure, the methanol hydroxy hydrogen forms a weaker hydrogen bond (2.45Å) with *O3*, which is also engaged in a hydrogen bonding interaction with acetic acid. In the context of the prototypical Fischer esterification mechanism, this difference in methanol orientation is critical. The first step in the Fischer esterification mechanism is the formation of a covalent bond between the alcohol oxygen and acid carboxyl carbon atoms[48]. For the *em-pam*$_2$ structure, the atoms are unobstructed at a distance of 2.76Å. For the *mm-pam*$_3$ structure, the atoms are 4.9Å apart and oriented away from each other (the carboxyl carbon faces the methanol methyl group). Therefore, the structure predicted with the *em* surface is more accessible for general Fischer-type esterification compared to the most similar *mm* structure.

While the *mm-pp* structures are obtained by optimization with the catalyst center SiO_3 coordinates frozen, no coordinates are frozen during optimization of *em-pp* structures. Instead, the *em-pp* structures are embedded in a bulk MSN surface which is simulated with a molecular mechanics force field. The optimizations of *em-pp* structures are geometrically constrained by interaction with the external forces of the bulk MSN surface. While the distance between the catalyst-anchoring silica atoms was fixed at 5.07Å for the *mm-pp* structures, the same Si-Si distance varies from 5.84-5.87Å among the optimized *em-pp* structures. The spatial flexibility of the propylsulfonic acid chain enables inter-catalyst hydrogen bonding interactions for a range of catalyst-anchoring Si-Si distances. Optimized without reactants present (*Figure 11*), the hydrogen bond distance between the catalyst sulfonyl groups is the same as for *mm-pp* (1.82Å). Although the catalyst-anchoring Si-Si distances are larger for the *em-pp* structure than for *mm-pp*, the propylsulfonic acid S-S distances are significantly shorter (4.21Å vs 4.87Å). The amorphous MSN surface is realistically represented with the *em* model, as the two SiO_3 groups anchoring the acid catalyst are oriented at different relative angles and surface depths. While the *mm-pp* and *em-pp* structures do exhibit some qualitative similarities, the reactant adsorption behavior differs significantly.

With the exception of the *mm-pmpa* structure, adsorbates on the *mm-pp* surface are generally localized to a single catalyst site in orientations similar to complexes found with the *mm-p* model. The steric accessibility of both *mm-pp* catalyst sites is nearly identical, and anchored at the same depth and orientation on the silica surface. Conversely, the inequality in steric accessibility of the two *em-pp* catalyst sulfonyl groups leads to delocalized adsorption across both catalysts.

Because of the relative orientations of the catalyst-anchoring SiO_3 groups, the catalyst sulfonyl groups are roughly perpendicular to one another for the *em-pp* structures. The relatively short distance between the MSN surface and the sulfonyl group perpendicular to the silica surface (pr_a) hinders the accessibility of the pr_a *O2* and *O3* atoms. The orientation of this sulfonyl group also exposes the adjacent alkane hydrogen atoms to the MSN pore. In comparison, the pr_d catalyst more closely resembles the *mm* catalyst sites. Hydrogen bonding between the catalyst sites and the reactants has been demonstrated to be the primary means of interaction during adsorption events. Clearly, variations in the number and morphology of potential hydrogen bonding sites as a result of the local MSN surface will significantly impact adsorption and reaction energetics.

Optimized *em-pp* structures with single adsorbates are shown in *Figure 12*. A hydrogen bond is formed between acetic acid and the pr_d *O1* in the *em-pap* structure. The acetic acid carbonyl group is rotated between both catalyst sulfonyl groups at a distance of 2.25Å from a pr_a alkane hydrogen atom and 2.40Å from the pr_d *O1* hydrogen. The acetic acid adsorption energy of 11.0 kcal mol^{-1} is less than the adsorption energy computed for the *mm-ppa* (18.7 kcal mol^{-1}) and *em-pa* (17.4 kcal mol^{-1}) structures. This may be attributed to the presence of only one hydrogen bond between the catalyst and acetic acid for the *em-pap* structure. Based on the O-H distance of the lone hydrogen bond, the interaction is also the weakest among those structures. The hydrogen bond length is 1.92Å, while the analogous hydrogen bond distances for the other structures range from 1.52-1.69Å. A hydrogen bond between a sulfonyl carbonyl oxygen and the methanol hydroxy hydrogen is exhibited in the *em-pmp* structure. The methanol orientation

resembles that of *mm-pm$_1$*, with methanol acting as the hydrogen donor in the hydrogen bond with the catalyst. The methanol adsorption energy for the *em-pmp* structure is 7.1 kcal mol^{-1} with a 2.03Å hydrogen bond length. This adsorption energy is comparable to the 5.2 kcal mol^{-1} value observed for the *mm-pm$_1$* structure (2.02Å hydrogen bond length). While the methanol methyl group of the *em-pmp* structure is rotated toward the neighboring catalyst site, the methanol adsorption observed is relatively localized to p_d.

Two *em-pp* structures with both reactants adsorbed were located (*Figure 13*). In the study of dual reactant adsorption with the *mm-pp* surface, structures were found for reactants adsorbed on separate catalyst sites, and on the same catalyst site. A search for analogous structures with the *em-pp* surface was unsuccessful. In both cases, acetic acid adsorption orientation is delocalized across both catalyst sites (denoted in the structure names by (a)). A hydrogen bond is present between the acetic acid hydroxy group and *pr$_d$ O1* in both structures. Based on the metric of the hydrogen bond distance, this hydrogen bond involving the acetic acid carbonyl group is weaker for the two-reactant structures (*em-ppm(a)*=2.17Å, *em-pmp(a)*=2.22Å) than the *em-pap* structure. The most significant difference between the dual-adsorbate *em-pp* structures is the position of methanol, which is adsorbed on *pr$_d$* in one structure, and on *pr$_a$* in the second structure. For the *em-ppm(a)* structure, methanol interacts more strongly with acetic acid than with the catalyst site. The methanol hydroxy hydrogen is oriented toward the acetic acid carbonyl oxygen at a bond length of 2.00Å. In comparison, the methanol hydroxy hydrogen is 2.32Å away from the closest sulfonyl oxygen atom. The *em-pmp(a)* structure exhibits a hydrogen bonding interaction between methanol and *pr$_d$ O2*, just as seen for the *em-pmp* structure. The *em-pmp(a)* hydrogen bond is slightly weaker (2.12Å vs 2.03Å). The methanol

methyl group is also rotated about the methanol O-H bond axis compared to *em-pmp*, oriented away from the acetic acid molecule. The methanol hydroxy oxygen may interact weakly through a hydrogen bond with an acetic acid methyl hydrogen with a distance of 2.34Å. The adsorption energies for both two-adsorbate *em-pp* structures are similar: 21.3 kcal mol^{-1} for *em-pmp(a)* and 20.6 kcal mol^{-1} for *em-ppm(a)*. The co-adsorption energy is notably weaker than was computed for the *mm-pp* dual-adsorbate structures (25.7-39.6 kcal mol^{-1}) and equivalent *em-pp* structures (23.6, 24.6 kcal mol^{-1}).

Analysis of Adsorption Results

For the reactants of interest, the single-catalyst molecular geometries and adsorption energies predicted by the minimal and embedded models are in agreement. The catalyst adsorption site is spatially separated from the MSN surface by the propyl chain, and the bulk effects of the local MSN surface do not seem to be particularly important. For the dual-site *mm,* both catalysts are nearly equivalent in terms of steric accessibility as the anchoring SiO_3 groups are at equal angle and depth relative to the plane of the silica surface. The adsorption energies of *mm-pp* single-adsorbate structures are in agreement with the analogous *mm-p* and *em-p* results. With the exception of the *mm-pmpa* catalyst-bridging methanol molecule, reactant adsorption on the *mm-pp* surface mimics that of the isolated single-catalyst systems. The *mm-pp* catalysts are both spatially accessible for adsorption, and are essentially equivalent to two isolated single-catalyst complexes with the exception of the inter-catalyst hydrogen bond. The *em-pp* structures represent the amorphous surface of an actual MSN system. The two catalyst sites are not equally accessible for surface adsorption, and neither site resembles an isolated single-catalyst site. Acetic acid adsorption is universally weaker for the *em-pp* structures compared to the other surface models, and is not strongly isolated to a single catalyst site. For the two instances of

methanol hydrogen bonding on *em-pp* catalyst sites, methanol acts as the hydrogen donor with adsorption energy similar to the analogous *mm* structures. The *em-pp* dual-adsorbate adsorption energies are lower than any of the dual-adsorbate adsorption energies computed for the other surface models.

Esterification Reaction Mechanisms

While the adsorption energies are sensitive to the catalytic surface, other important factors for initiation of the esterification reaction include the relative location and orientations of the adsorbed species. The initial step of the general Fischer esterification reaction is protonation of the carboxylic acid reactant by the acid catalyst. Therefore, acetic acid must be oriented such that the carbonyl oxygen can be protonated or activated by hydrogen bonding with propylsulfonic acid. Based on the *em-pp* results, the relative orientations of catalyst sites are shown to have a significant effect on adsorption energies and adsorbate orientations. In order to separate this work from the discussion of the relative orientation of catalyst species, a single-catalyst surface is used to investigate the reaction mechanism. For other studies in which dual-catalyst mechanisms are proposed for the reaction of interest, the catalyst site with adsorbed methanol serves primarily to mediate methanol diffusion to an adjacent catalyst site with pre-adsorbed acetic acid[10]. The esterification reaction then proceeds at a single catalyst site, which is the main focus in this work. Two potential reaction mechanisms are presented. In *Scheme 1*, a methanol molecule interacts directly with a pre-adsorbed acetic acid (ER-type) molecule. In *Scheme 2*, both reactants are initially co-adsorbed and non-interacting (LH-type). All reaction mechanism structures were located with the single-catalyst *em* surface.

The optimized structures for the *Scheme 1* pathway are shown in *Figure 14*. The initial structure A is *em-pam*$_2$, in which methanol is positioned above acetic acid without strong interactions with the catalyst surface. Proceeding through TS1, the acetic acid molecule is rotated about the hydrogen bond with *O1*, disrupting the hydrogen bond with *O3*. A covalent bond is formed between the methanol oxygen and carboxyl carbon, and the methanol hydroxy hydrogen is transferred directly to the acetic acid carbonyl oxygen. Following TS1, it is proposed that the reaction proceeds through structure B. Notably, a transition state (with a barrier of 8.8 kcal mol^{-1}) connected to B was located in which the reactant hydroxy group hydrogen bonded to the catalyst protonates *O2* while the catalyst protonates the same oxygen site. A structure (B_1) resembling the B intermediate was also located with the *em-pp* surface. B_1 is higher in energy than the dual-adsorbate *em-pp* structures by an amount that is in sub-kcal mol^{-1} agreement with the difference between A and B. Structure B rotates about the carboxyl-hydroxy bond that is hydrogen bonded to the catalyst to reach C. Structure C is 3.4 kcal mol^{-1} lower in energy than B, and exhibits an additional hydrogen bond with the catalyst *O3* site which is present in the subsequent structures. The forward reaction proceeds through TS2 with a barrier of 11.0 kcal mol^{-1}. In TS2 the hydroxy group formed through TS1 bonds to the sulfonyl *O1* hydrogen while the ester intermediate protonates the catalyst sulfonyl group. The final product structure D is 2.9 kcal mol^{-1} lower in energy than A. The highest energy barrier for the forward reaction in *Scheme 1* (40.6 kcal mol^{-1}) is greater than the highest energy barrier for the reverse ester hydrolysis reaction (32.1 kcal mol^{-1}).

The optimized structures for the *Scheme 2* pathway are shown in *Figure 15*. Unlike *Scheme 1*, all structures in *Scheme 2* exhibit two hydrogen bonds between the reactants and catalyst surface. Initial structure A' is *em-pam*$_1$, which represents stepwise co-adsorption of the adsorbates. Through the first transition state TS1', the methanol oxygen and carboxyl carbon atoms form a covalent bond. The *O2* oxygen with respect to structure A' is protonated by methanol, while the acetic acid carbonyl oxygen is simultaneously protonated by the *O1* hydrogen. This proton exchange pathway is significantly different from *Scheme 1*, in which the acetic acid carbonyl oxygen is directly protonated by methanol. The A'→TS1' energy barrier of 12.9 kcal mol^{-1} is decidedly smaller than the direct methanol protonation barrier as well. Structure B' is just 4.3 kcal/mol^{-1} higher in energy than A'. Considering the relatively low energy barrier though TS1', interconversion between A' and B' is highly reversible. The hydrogen bond between the oxygen from the methanol oxygen and the catalyst is shifted to *O3* in the structures located for TS2' and C', compared to *O2* for the first three structures. A relatively large barrier height of 40.6 kcal mol^{-1} is associated with the B'→TS2' transition. The high energy of TS2' is due to the formation of a strained four-member ring structure. A similar structure was proposed in a previous work investigating the esterification mechanism between acetic acid and ethanol[26], but the transition state is stabilized by only one hydrogen bond in that study. The product structure C is 1.2 kcal mol^{-1} lower in energy than A. The forward reaction is energetically favorable relative to the reverse reaction, with a barrier height of 40.6 kcal mol^{-1} compared to 46.0 kcal mol^{-1} for ester hydrolysis. MP2 predicts larger reaction barriers than DFT in this work. Many functionals are well known to underestimate the heights of reaction barriers.

Conclusions

The esterification of acetic acid with methanol by propylsulfonic acid-functionalized silica was investigated in this work. The adsorption behavior of acetic acid and methanol across single and dual catalyst sites was investigated on both simple silica surfaces, and on an extended MSN surface. It was determined that the single-catalyst adsorbate orientation and adsorption energies on the *mm*-type surface are in agreement with the more realistic and more computationally expensive *em*-type surface. The same results for single-adsorbate dual-catalyst structures are in agreement as well. The differences between the dual-adsorbate dual-catalyst computations are attributed not to the differences in the silica support structure, but rather the relative orientation of the catalyst pairs. The amorphous surface of silica captured by the *em* model offers the potential to sample many catalyst pair orientations. These orientations have a measured effect on adsorption activity, and thus the subsequent esterification reaction.

Two stepwise reaction mechanisms were proposed with the *em* single-catalyst model. Both reactions begin with protonation of acetic acid. In the first scheme, pre-adsorbed acetic acid is protonated directly by methanol. The transition state corresponding to the initial protonation step is the highest energy structure, and the highest energy barrier for the forward is 40.6 kcal mol^{-1}, compared to 32.1 kcal mol^{-1} for the reverse reaction. In the second scheme, acetic acid and methanol are co-adsorbed on adjacent sites of the catalyst. Acetic acid protonation by the catalyst is coupled with protonation of the catalyst by methanol. The highest energy reaction barrier for the forward reaction is the same as in the first scheme (40.6 kcal mol^{-1}), and is favored over ester hydrolysis by 5.5 kcal mol^{-1}. Both schemes are reversible and energetically similar, so the chosen

reaction pathway is proposed to depend primarily on the whether or not both reactants are initially co-adsorbed. For both pathways, the highest energy transition state structures exhibit strained 4-member rings. In a separate study[49], a reaction pathway involving a 4-member ring transition state was converted to a lower energy pathway with a 6-member transition state following the addition of a water molecule. Although the presence of water is detrimental to the esterification reaction yield because of ester hydrolysis, the addition of explicit aqueous solvent effects may reveal a similar transition state stabilization effect.

While the initial interaction of reactants may be mediated by diffusion following reactant adsorption on adjacent catalyst sites, adsorption on dual-catalyst sites is highly sensitive to the variability in relative catalyst orientations on the amorphous MSN surface. Further investigation of the relationship between relative catalyst orientations and adsorption behavior on the MSN surface with the *em-pp* surface is a current research effort.

Acknowledgments

This work was supported in part by a Basic Research Initiative grant from the Air Force Office of Scientific Research (Award Number FA9550- 12-1-0476)

References

1. Gui, M.; Lee, K.; Bhatia, S. *Energy* **2008**, *33*(11), 1646.
2. Carmo, A. C. A. D. C.; Souza, L. K. D.; Costa, C. E. D.; Longo, E.; Zamian, J. R.; Filho, G. N. D. R. *Fuel* **2009**, *88*(3), 461.
3. Mbaraka, I.; Shanks, B. *J. Catal.* **2005**, *22* (2), 365.
4. Mbaraka, I.; Radu, R. R.; Lin, V.; Shanks, B. *J. Catal.* **2003**, *219* (2), 329.
5. Mbaraka, I. K.; Mcguire, K. J.; Shanks, B. H. *Ind. Eng. Chem. Res.* **2006**, *45*(9), 3022.
6. Gu, G.; Ong, P.; Chu, C. *J. Phys. Chem. Solids* **1999**, *60*(7), 943.
7. Beck, J. S.; Vartuli, J. C.; Roth, W. J.; Leonowicz, M. E.; Kresge, C. T.; Schmitt, K. D.; Chu, C. T. W.; Olson, D. H.; Sheppard, E. W.; Mccullen, S. B.; Higgins, J. B.; Schlenker, J. L. *J. Am. Chem. Soc.* **1992**, *114*(27), 10834.
8. Zhao, D.; Huo, Q.; Feng, J.; Chmelka, B.; Stucky, G. *J. Am. Chem. Soc.* **1998**, *120(24)*, 6024.
9. Teo, H.; Saha, B. *J. Catal.* **2004**, *228*(1), 174.
10. Miao, S.; Shanks, B. H. *J. Catal.* **2011**, *279*(1), 136.
11. Liu, Y.; Lotero, E.; Goodwinjr, J. *J. Catal.* **2006**, *242*(2), 278.
12. Koster, R.; Linden, B. V. D.; Poels, E.; Bliek, A. *J. Catal.* **2001**, *204*(2), 333.
13. Jermy, B. R.; Pandurangan, A. *Appl. Catal. A Gen.* **2005**, *288*(1-2), 25.
14. Langmuir, I. *J. Am. Chem. Soc.* **1918**, *40*(9), 1361.
15. Hinshelwood, C. *Nature* **1930**, *125*(3158), 703.
16. Eley, D. D.; Rideal, E. K. *Proc. R. Soc. Lond. A Math Phys. Sci.* **1941**, *178*(975), 429.
17. Eley, D. D.; Rideal, E. K. *Nature* **1940**, *146*, 401.

18. Suwannakarn, K.; Lotero, E.; Ngaosuwan, K.; Goodwin, J. G. *Ind. Eng. Chem. Res.* **2009**, *48*(6), 2810.

19. Nijhuis, T.; Beers, A.; Kapteijn, F.; Moulijn, J. *Chem. Eng. Sci.* **2002**, *57* (9), 1627.

20. Chu, W.; Yang, X.; Ye, X.; Wu, Y. *Appl. Catal. A Gen.* **1996**, *145*(1-2), 125.

21. Batista, A. P. D. L.; Zahariev, F.; Slowing, I. I.; Braga, A. A. C.; Ornellas, F. R.; Gordon, M. S. *J. Phys. Chem. B* **2016**, *120*(8), 1660.

22. Folliet, N.; Gervais, C.; Costa, D.; Laurent, G.; Babonneau, F.; Stievano, L.; Lambert, J.-F.; Tielens, F. *J. Phys. Chem. C* **2013**, *117*(8), 4104.

23. Tielens, F.; Folliet, N.; Bondaz, L.; Etemovic, S.; Babonneau, F.; Gervais, C.; Azaïs, T. *J. Phys. Chem. C* **2017**, *121*(32), 17339.

24. Ravikovitch, P. I.; Neimark, A. V. *Langmuir* **2006**, *22*(26), 11171.

25. Fujiki, J.; Furuya, E. *Fuel* **2016**, *164*, 180.

26. Vafaeezadeh, M.; Fattahi, A. *Comp. Theor. Chem.* **2015**, *1071*, 27.

27. Lee, C.; Yang, W.; Parr, R. G. *Phys. Rev. B* **1988**, *37*(2), 785.

28. Becke, A. D. *J. Chem. Phys.* **1993**, *98*(7), 5648.

29. Frondel, C. *Am. Mineral.* **1979**, *64*, 799.

30. Roskop, L.; Fedorov, D. G.; Gordon, M. S. *Mol. Phys.* **2013**, *111*(9-11), 1622.

31. Maseras, F.; Morokuma, K. *C. Comput. Chem.* **1995**, *16*(9), 1170.

32. Shoemaker, J. R.; Gordon, M. S. *J. Phys. Chem. A* **1999**, *103*(17), 3245.

33. Allinger, N. L.; Yuh, Y. H.; Lii, J. H. *J. Am. Chem. Soc.* **1989**, *111*(23), 8551.

34. Lii, J. H.; Allinger, N. L. *J. Am. Chem. Soc.* **1989**, *111*(23), 8576.

35. Kitaura, K.; Ikeo, E.; Asada, T.; Nakano, T.; Uebayasi, M. *Chem. Phys. Lett.* **1999**, *313*(3-4), 701.

36. Ponder, J. W.; Richards, F. M. *J. Comput. Chem.* **1987**, *8*(7), 1016.

37. Kundrot, C. E.; Ponder, J. W.; Richards, F. M. *J. Comput. Chem.* **1991**, *12*(3), 402.

38. Zhao, Y.; Truhlar, D. G. *Theor. Chem. Acc.* **2007**, *120*(1-3), 215.

39. Hehre, W. J.; Ditchfield, R.; Pople, J. A. *J. Chem. Phys.* **1972**, *56*(5), 2257.

40. Francl, M. M.; Pietro, W. J.; Hehre, W. J.; Binkley, J. S.; Gordon, M. S.; Defrees, D. J.; Pople, J. A. *J. Chem. Phys.* **1982**, *77*(7), 3654.

41. Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993**, *14*(11), 1347.

42. Gordon, M. S.; Schmidt, M. W. *In theory and applications of computational chemistry: The first forty years*; Elsevier: Amsterdam, The Netherlands, **2005**.

43. Chemcraft. https://chemcraftprog.com. (accessed November, 2016)

44. Zhuravlev, L. T. *Langmuir* **1987**, *3*(3), 316.

45. Batonneau-Gener, I.; Yonli, A.; Trouvé, A.; Mignard, S.; Guidotti, M.; Sgobba, M. *Sep. Sci. Technol.* **2010**, *45*(6), 768.

46. Musso, G. E.; Bottinelli, E.; Celi, L.; Magnacca, G.; Berlier, G. *Phys. Chem. Chem. Phys.* **2015**, *17*(21), 13882.

47. Yang, D.; Xu, Y.; Wu, D.; Sun, Y.; Zhu, H.; Deng, F. *J. Phys. Chem. C* **2007**, *111*(2), 999.

48. Fischer, E.; Speier, A. *Untersuchungen aus Verschiedenen Gebieten* **1924**, 285.

49. Sok, S.; Gordon, M.S. *Comp. Theor. Chem.* **2012**, *987*, 2.

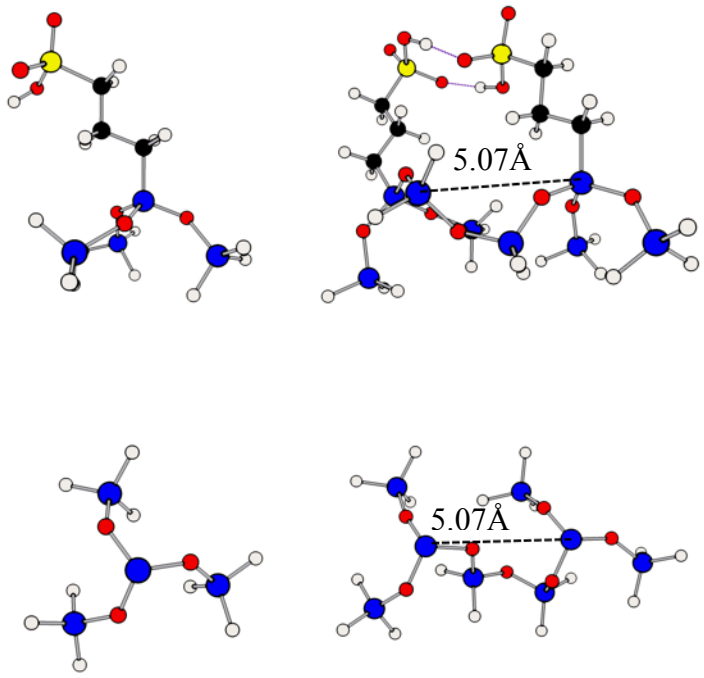
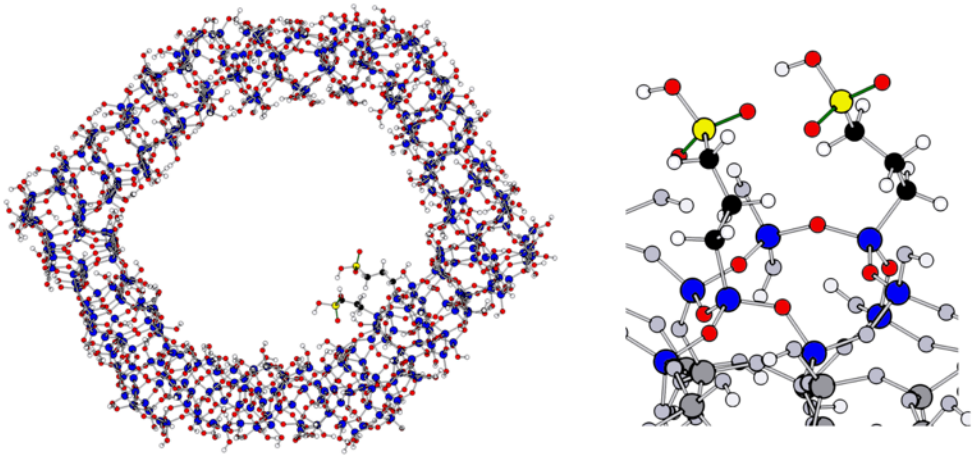


Figure 1. Minimal model silica structures for (clockwise from top-left) acid-functionalized single-site, acid-functionalized dual-site, and the isolated silica structures which support the dual site, and single-site models. H atoms are white, C atoms are black, Si atoms are blue, O atoms are red, and S atoms are yellow.



(A) (B)

Figure 2. (A) Mesoporous silica pore system, and (B) embedded acid-functionalized dual-catalyst system. Atoms in (B) treated with MM methods are grey, and atoms treated with QM methods are displayed in color. The color scheme is the same as that in *Figure 1*.

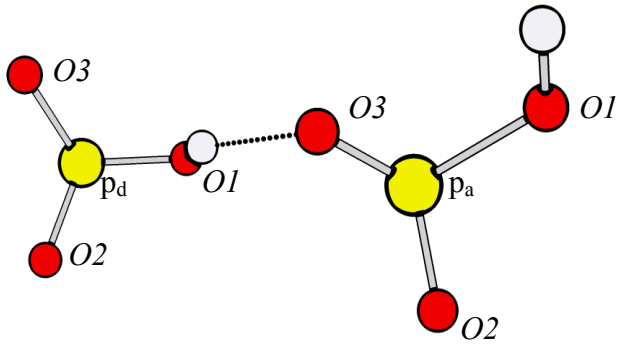
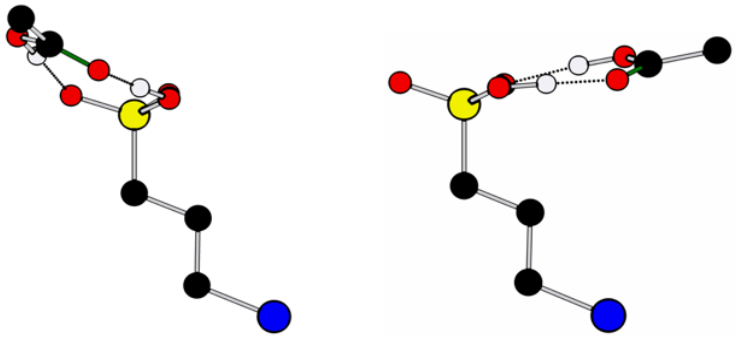


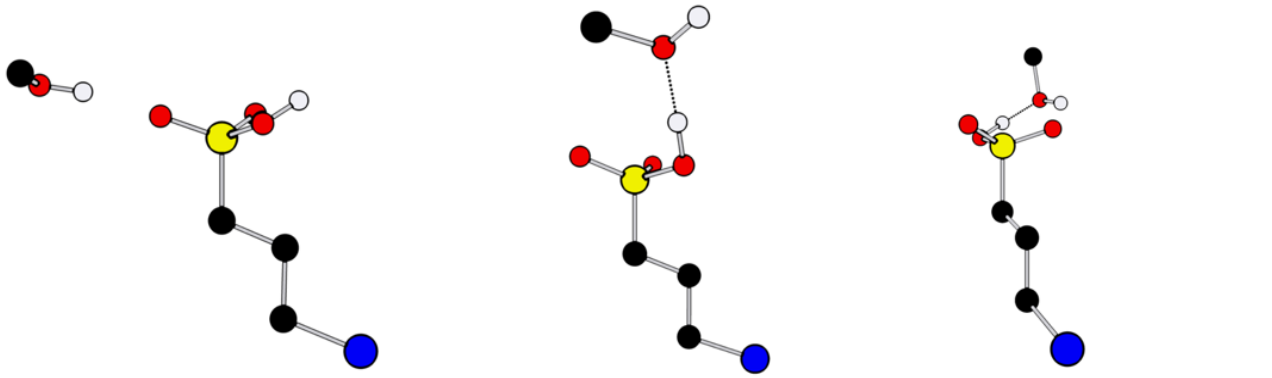
Figure 3. Diagram of atom nomenclature for catalyst sulfonyl atoms.



mm-pa$_1$
E_{ads} = 17.9 (19.3) kcal mol^{-1}

mm-pa$_2$
E_{ads} = 20.2 (20.8) kcal mol^{-1}

Figure 4. Optimized structures for acetic acid adsorption on the single-catalyst minimal model surface. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

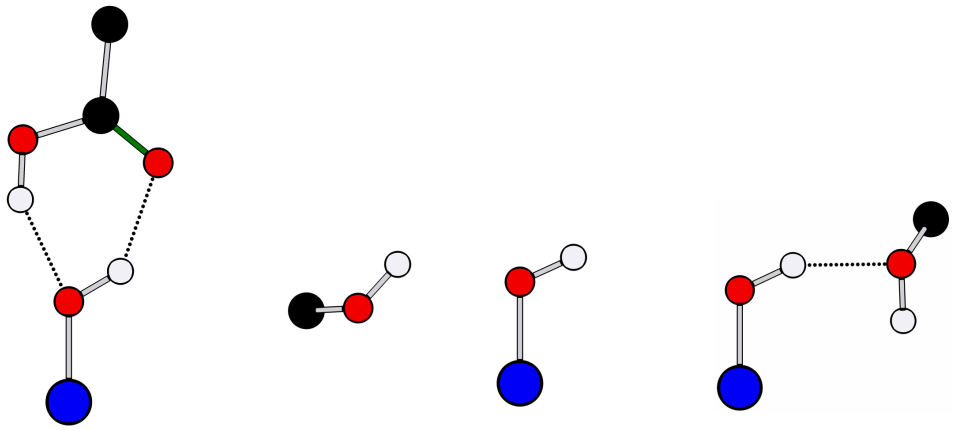


mm-pm$_1$
E_{ads} = 5.6 (5.2) kcal mol^{-1}

mm-pm$_2$
E_{ads} = 12.6 (12.3) kcal mol^{-1}

mm-pm$_3$
E_{ads} = 14.3 (15.2) kcal mol^{-1}

Figure 5. Optimized structures for methanol adsorption on the single-catalyst minimal model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

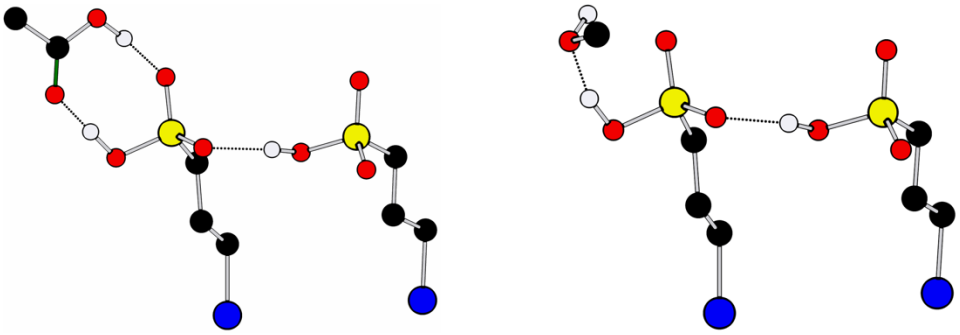


mm-sa
E_{ads} = 10.7 (11.5) kcal mol^{-1}

mm-sm$_1$
E_{ads} = 9.7 (10.4) kcal mol^{-1}

mm-sm$_2$
E_{ads} = 9.9 (11.6) kcal mol^{-1}

Figure 6. Optimized structures for acetic acid and methanol adsorption on the silanol minimal model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.



mm-ppa
E_{ads} = 18.7 (19.9) kcal mol^{-1}

mm-ppm
E_{ads} = 15.4 (16.4) kcal mol^{-1}

Figure 7. Optimized structures for acetic acid (left) and methanol (right) adsorption on the dual-catalyst minimal model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

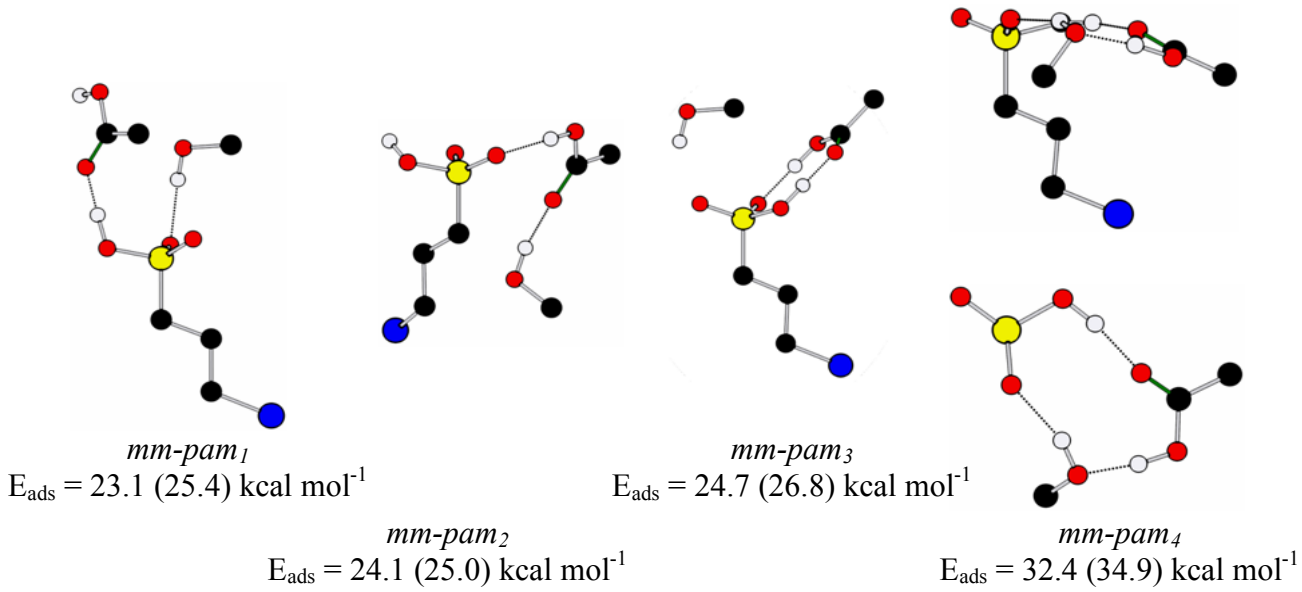


Figure 8. Optimized structures for acetic acid and methanol co-adsorption on the single-catalyst minimal model surface. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

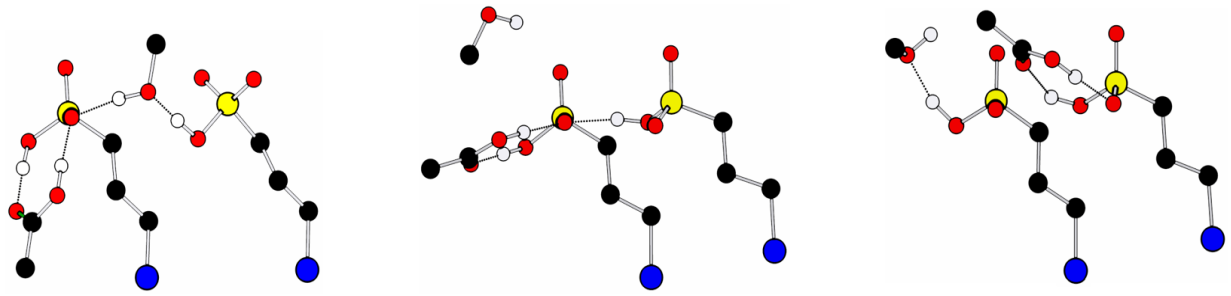
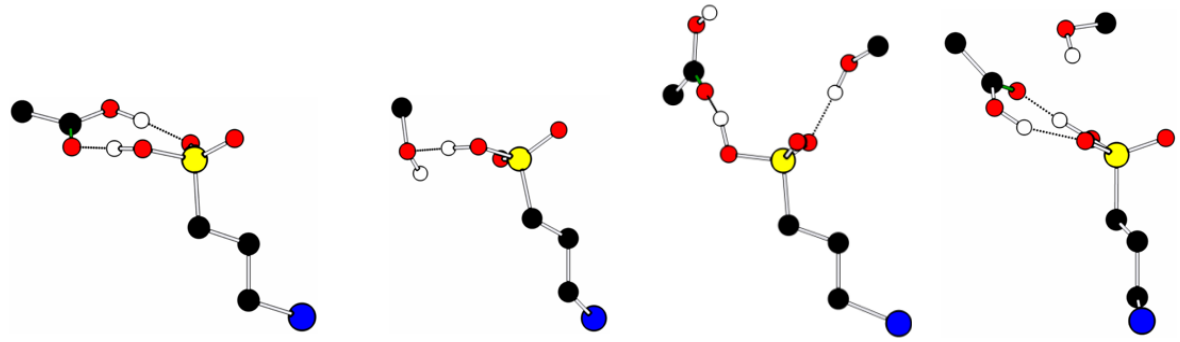


Figure 9. Optimized structures for acetic acid and methanol co-adsorption on the dual-catalyst minimal model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.



em-pa
E_{ads} = 17.4 (18.7) kcal mol^{-1}

em-pam$_1$
E_{ads} = 23.6 (24.8) kcal mol^{-1}

em-pm
E_{ads} = 12.2 (13.0) kcal mol^{-1}

em-pam$_2$
E_{ads} = 24.6 (26.8) kcal mol^{-1}

Figure 10. Optimized structures for acetic acid and methanol adsorption and co-adsorption on the single-catalyst embedded model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

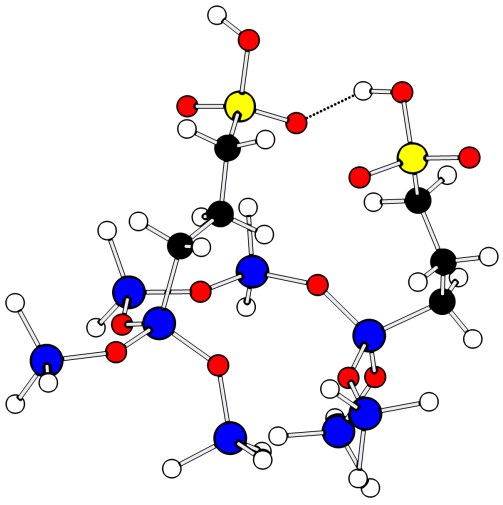
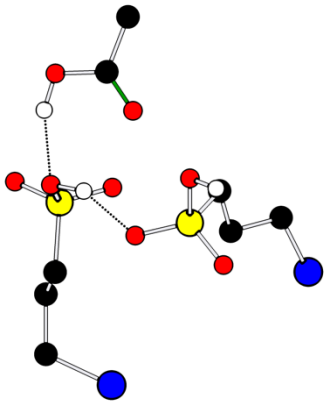
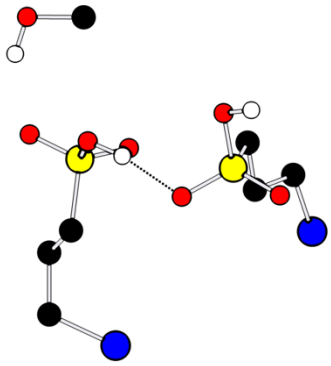


Figure 11. Optimized structure of dual-catalyst embedded model surface. The color scheme is the same as that in *Figure 1*.

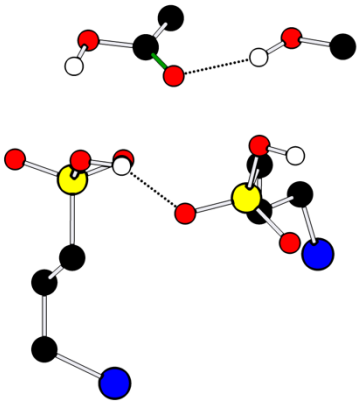


em-pap
E_{ads} = 11.0 (9.9) kcal mol^{-1}

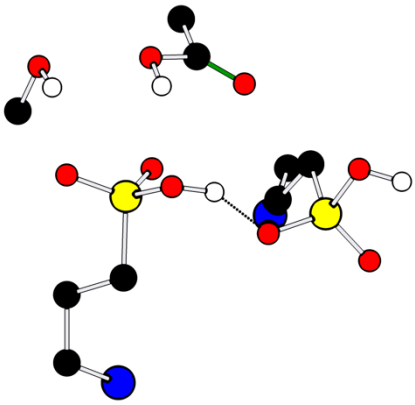


em-pmp
E_{ads} = 7.1 (6.7) kcal mol^{-1}

Figure 12. Optimized structures for acetic acid and methanol adsorption on the dual-catalyst embedded model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.



em-ppm(a)
E_{ads} = 20.6 (20.7) kcal mol^{-1}



em-pmp(a)
E_{ads} = 21.3 (19.3) kcal mol^{-1}

Figure 13. Optimized structures for acetic acid and methanol co-adsorption on the dual-catalyst embedded model surface. The methyl hydrogens are not shown. The adsorption energies shown were computed with MP2 (DFT). The color scheme is the same as that in *Figure 1*.

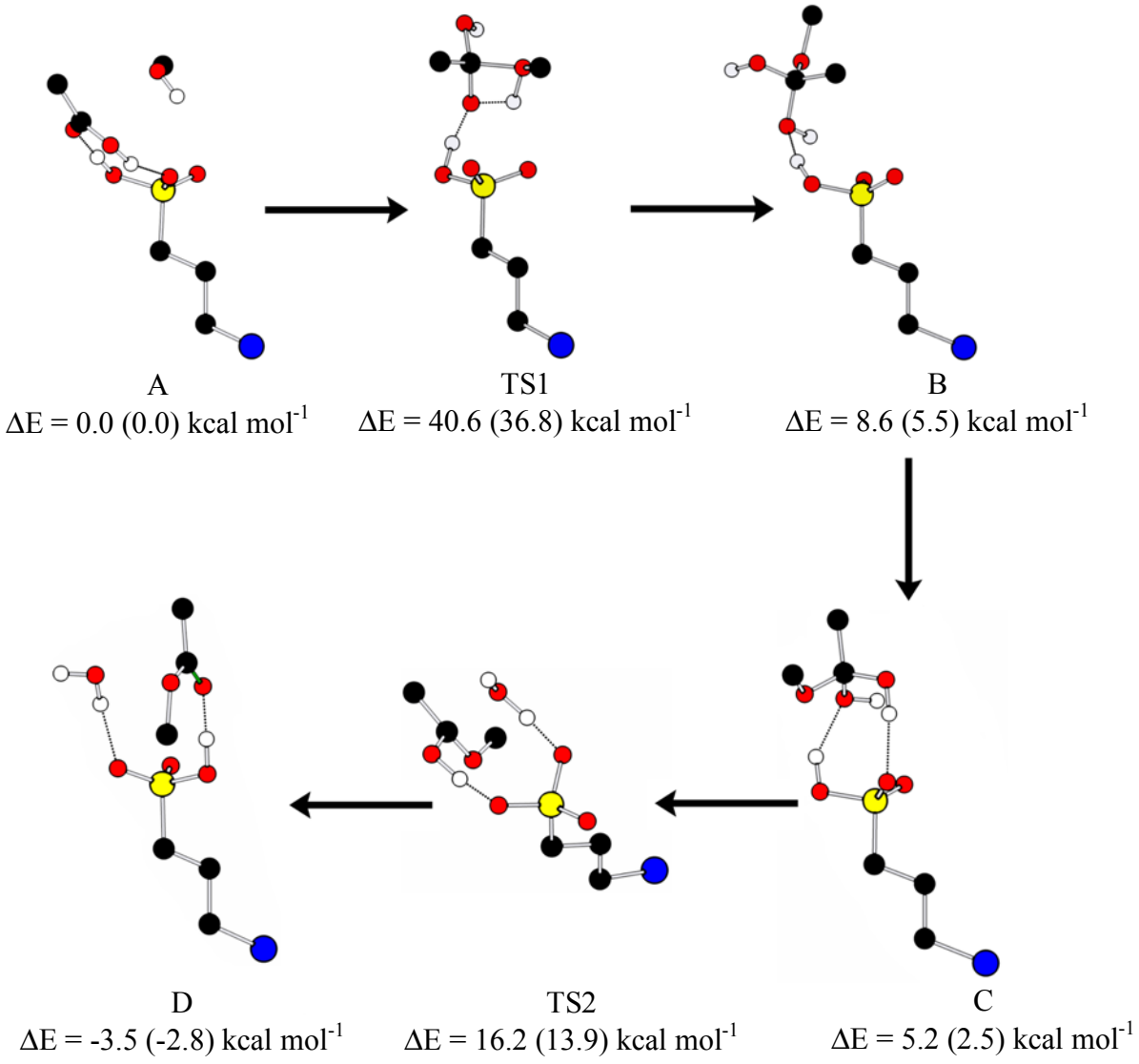


Figure 14. Proposed esterification reaction mechanism for single-catalyst surface, *Scheme 1*. The methyl hydrogens are not shown. ΔE is relative to the MP2 (DFT) energy of structure A. The color scheme is the same as that in *Figure 1*.

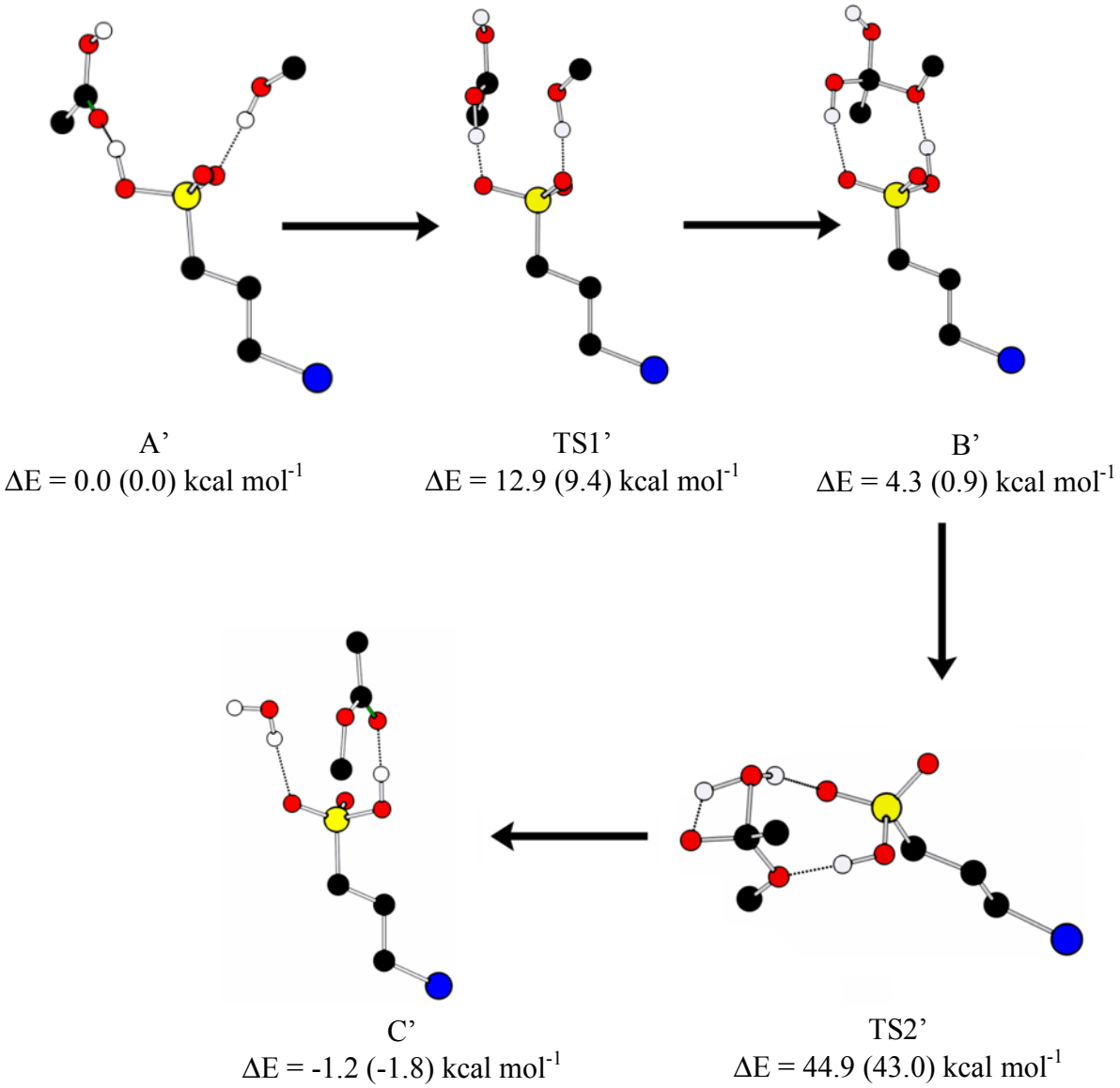


Figure 15. Proposed esterification reaction mechanism for single-catalyst surface, *Scheme 2*. The methyl hydrogens are not shown. ΔE is relative to the MP2 (DFT) energy of structure A’. The color scheme is the same as that in *Figure 1*.

CHAPTER 7: DYNAMICS SIMULATIONS WITH SPIN-FLIP TIME-DEPENDENT DENSITY FUNCTIONAL THEORY: PHOTOISOMERIZATION AND PHOTOCYCLIZATION MECHANISMS OF *cis*-STILBENE IN ππ* STATES

A paper published in the *Journal of Physical Chemistry A*

Yu Harabuchi, Kristopher Keipert, Federico Zahariev, Tetsuya Taketsugu, and Mark S. Gordon

Abstract

On-the-fly dynamics simulations were carried out using spin-flip time dependent density functional theory (SF-TDDFT) to examine the photoisomerization and photocyclization mechanisms of *cis*-stilbene following excitation to the ππ* state. A state tracking method was devised to follow the target state among nearly degenerate electronic states during the dynamics simulations. The steepest descent path from the Franck-Condon structure of *cis*-stilbene in the ππ* state is shown to reach the S_1-minimum of 4,4-dihydrophenanthrene (DHP) *via* a *cis*-stilbene-like structure (referred to as $(S_1)_{cis\text{-min}}$) on a very flat region of the S_1-potential energy surface. From the dynamics simulations, the branching ratio of the photoisomerization is calculated as *trans*: DHP = 35: 13, in very good agreement with the experimental data, *trans*: DHP = 35: 10. The discrepancy between the steepest descent pathway and the significant *trans*-stilbene presence in the branching ratio observed experimentally and herein computationally is clarified from an analysis of geometrical features along the reaction pathway, as well as the low barrier of 0.1 eV for the pathway from $(S_1)_{cis\text{-min}}$ to the twisted pyramidal structure on the S_1-

potential energy surface. It is concluded that ππ*-excited *cis*-stilbene propagates primarily toward the twisted structural region due to dynamic effects, with partial branching to the DHP structural region *via* the flat-surface region around $(S_1)_{cis\text{-min}}$.

Introduction

1,2-diphenylethylene (stilbene) and its derivatives play many roles in both science and everyday life. The combination of tunable photophysical properties and high thermal and chemical stability have led to a wide variety of research and practical applications of stilbenoids. Stilbene and its derivatives are commonly used as optical brightening agents, molecular probes, and gain mediums in blue dye lasers for both spectroscopic and laser medicine purposes. Stilbenoids absorb light in the UV (usually 340-370 nm) region, and re-emit light in the blue region (typically 420-470 nm). These tunable photophysical properties have been utilized to make stilbenoid whitening agents, which enhance the appearance of fabric and paper by causing a "whitening" effect. These additives make materials look less yellow by increasing the relative amount of blue light reflected. Stilbenoids are naturally present in some plants, such as the presence of the phytoalexin agent 3,5,4'-trihydroxystilbene (reservatrol) in groundnuts, raspberries and blackberries. Humans have taken advantage of stilbenoid biological properties to produce medicine, such as the traditional "Puag-Haad" natural extract, which contains an abundance of oxyresveratrol, used to treat tapeworm infections in Southeast Asia.

Stilbene is a widely studied molecule that undergoes photoisomerization.[1-3] There are two isomers in the ground state of stilbene, *i.e. cis* and *trans*-forms, *cis*-stilbene can undergo

photocyclization to produce 4,4-dihydrophenanthrene (DHP). Due to interest in the reaction mechanism of photo-excited molecules, the photoisomerization mechanism of *cis*-stilbene has been examined extensively both experimentally[4-36] and theoretically.[31,36-50] It is known[5,7,13,15,17] that photo-excited *cis*-stilbene can lead to all three of the structures as shown in *Figure 1*. In early experimental research, the quantum yield for the photoreaction of the ππ* excitation of *cis*-stilbene is reported to be 0.10 for DHP, 0.35 for *trans*-stilbene, and 0.55 for the *cis*-stilbene.[5,7,13] The observation of three photoreaction products indicates that reaction channels for both *trans*-stilbene and DHP are open in the ππ* state of *cis*-stilbene. Experimental time-resolved studies report that the lifetime of *cis*-stilbene after excitation to the ππ* state is very short,[8,10,12,14,18-21,24-26,28,29,32,33,35] and that the photoreaction process proceeds rapidly following photoexcitation. The ring formation mechanism of *cis*-stilbene, *i.e.*, cyclization to DHP, is important when considering diarylethene derivatives for photo-switching molecule applications.[51-53] Several experimental studies have examined the *cis*-DHP cyclization of stilbene.[4,5,7,13,15,23]

Two important optical experiments were performed very recently by Tahara and co-workers in which both time resolved Raman spectra[31] and femtosecond time-resolved fluorescence spectra were performed.[35] From the Raman spectra, it is shown that stilbene exhibits a vibrational mode frequency downshift from 239 cm^{-1} (0.3 ps) to 224 cm^{-1} (1.2 ps) to 215 cm^{-1} (2.0 ps) after an initial upshift from 231 cm^{-1} (0.0 ps) to 239 cm^{-1} (0.3 ps). The authors assign the vibrational mode to twisting about the C=C bond based on both the steepest descent path in the excited state and vibrational analyses along the path. They conclude that the gradual downshift of the experimental frequency corresponds to gradual twisting of the central C=C bond which accompanies the pyramidalization of carbon atoms in the central C=C bond. Based

on the fluorescence spectra,[35] it is shown that the decay of *cis*-stilbene is bi-exponential, corresponding to a 0.23 ps fast process and a 1.2 ps slow process with oscillator strengths of 0.32 and 0.21 respectively. The fast fluorescence component (0.23 ps) is attributed by the authors to the cis^*_A state which is reached just prior to the twisting of the C=C bond, but after the initial elongation of the central C=C bond. The slow fluorescence component (1.2 ps) is attributed to the cis^*_B state in which the central C=C bond has twisted substantially around a shallow S_1 potential energy minimum. In addition, it was concluded that the observed oscillator strengths indicate that the population branching selection to either *trans*-stilbene or DHP occurs in a very early stage of the photoreaction process of *cis*-stilbene.

The *cis-trans* photoisomerization process has been widely examined in many theoretical studies.[31,37-50] A major focus has been to clarify both geometric and electronic structures along the relaxation paths including the conical intersection (CI) region corresponding to the decay channel from the excited state to the ground state (S_1/S_0).[38,40,45,47,49,50] It is generally accepted that the relaxation of *cis*-stilbene in the ππ* state involves a twisting motion about the central C=C bond,[37,38,40,45,47,49,50] and that the S_1/S_0-CI region exists near the twisted minimum of the ππ* state.[38,40,45,47,49,50] The molecular motion of the *cis-trans* photoisomerization has been predicted to be a "hula-twist" motion, in which the central C=C bond rotates out of plane; the C-H bonds remain out of the plane, while the other atoms reorient to remain coplanar.[27] With regard to the cyclization process from *cis*-stilbene to DHP (denoted *cis*-DHP), Bearpark was the first to report the location of the conical intersection corresponding to DHP formation with a hybrid molecular mechanics-valence bond method (MMVB).[38] Very recently, the detailed potential energy surfaces (PES) for both photoreactions of *cis*-stilbene were examined using spin-flip (SF) time-

dependent density functional theory (TDDFT)[47] and extended multiconfiguration quasi-degenerate second order perturbation theory (XMCQDPT2).[50] It was reported that the twisting motion about the central C=C bond, *i.e. cis-trans* isomerization, appears to be preferred in a ππ* state of *cis*-stilbene, rather than the ring closing motion, *i.e. cis*-DHP cyclization. One MMVB study[38] draws the opposite conclusion.

There have been two reports of dynamics simulations of the photoreaction in *cis*-stilbene. Berweger employed singly excited configuration interaction (CIS) with the 6-31G basis set to study dynamics simulations on a three-dimensional constrained PES which was obtained by an interpolation scheme.[39] The authors were unable to consider the S_1/S_0-crossing regions due to their limited dimensional PES. However, they reported that the dihedral angle of the central C=C torsion changes from 0° to 180° along the trajectories in the stilbene excited state, corresponding to the *cis-trans* isomerization. Dou *et al.*[43,44,46] performed a semi-classical electron-radiation-ion dynamics (SERID) simulation on the excited state of *cis*-stilbene. They reported three different trajectories in three papers, *cis-trans* isomerization,[44] *cis-cis* (no isomerization),[43] and *cis*-DHP cyclization.[46] Based on the *cis-trans* isomerization trajectory,[44] they reported that the excitation from the highest occupied molecular orbital (HOMO) to the lowest unoccupied molecular orbital (LUMO) of *cis*-stilbene rapidly leads to the formation of DHP at 0.2 ps, and that a subsequent *cis-trans* photoisomerization occurs *via* a twisted pyramidal CI which has an intermediate dihedral angle (90°) about the central C=C bond. For the *cis-cis* trajectory,[43] they reported that stilbene passes through twisted pyramidal CI regions which are similar to those in the *cis-trans* photoisomerization. For the *cis*-DHP trajectory,[46] they proposed that a new chemical bond is formed between two phenyl rings at 0.6 ps *via* the *trans*-isomer (dihedral angle = 180° about the

central C=C bond). These reports by Dou *et al.* indicate very interesting stilbene phenomena, since DHP-*trans* and *trans*-DHP reactions in the excited state were observed in the trajectories of the *cis*-*trans* and *cis*-DHP reactions, respectively. To date there has been no report of *ab initio* dynamics simulations dealing with full dimensional motions for *cis*-stilbene, possibly due to both the prohibitive computational cost and the difficulties of accurately describing the CI region discussed below. There has also been no report about the photoreaction branching mechanism, *i.e.* the comparison of trajectories, between *cis*-*trans* and *cis*-DHP in the $\pi\pi^*$ state.

The main focus of the present study is to clarify the photoreaction branching mechanism of stilbene from *cis*-stilbene to both *trans*-stilbene and DHP by utilizing on-the-fly dynamics simulations. Also, the time scale of the photoreaction is examined based on comparisons with experimental results, and the origin of the experimental spectra is discussed by considering the predicted molecular motions.

Dynamics simulations have become a powerful tool for examining the mechanisms of chemical reactions. Computational advances now allow for practical on-the-fly dynamics simulations for photochemical reactions that occur in the sub-picosecond timeframe. Such simulations can provide insights that cannot currently be obtained experimentally. These dynamics simulations generally use classical trajectories in combination with some level of electronic structure theory. Many chemical reaction mechanisms have been clarified using on-the-fly dynamics simulations.[54-56]

When studying photochemical reactions with dynamics simulations, there are several issues that occur in the study of non-adiabatic transitions through CI regions.[57-64] One issue is

how to deal with intersystem crossing in non-adiabatic regions. Since non-adiabatic transitions cannot be considered within the Born-Oppenheimer approximation, wavepacket methods or semi-classical approximations must be employed to treat non-adiabatic transitions. The other difficulty is that multi-configurational effects and dynamic correlation are usually necessary to correctly and quantitatively describe excited state PESs in CI regions. Thus, the computational methods employed must be chosen carefully to accurately describe the excited state dynamics.

For dynamics simulations in excited states, multi-configurational methods such as the complete active space self-consistent field (CASSCF) method are commonly used to describe CI regions. To include dynamic correlation, one must build upon the CASSCF wavefunction in the form of multi-reference perturbation theory (MRPT),[65-67] or multi-reference configuration interaction (MRCI).[68,69] This requirement can be a significant obstacle when performing excited state dynamics, because application of multi-reference methods can have prohibitively high computational cost, and because of the possibility that the most appropriate active space may change along a PES. Analytic gradients are not universally available for post-CASSCF methods, which further contributes to the high computational cost of including dynamic correlation.

In the present study, time dependent density functional theory (TDDFT) is used for the excited-state dynamics simulations. TDDFT often provides reasonable results for excited electronic states, at a relatively low cost. TDDFT has been employed extensively to describe PESs of excited states, and most functionals incorporate dynamic correlation. The computational cost of TDDFT is much lower than that of multi-reference methods, such as MRPT and MRCI, especially for large molecules.[70-72] TDDFT also avoids the active space decisions that are

frequently necessary for multi-reference methods. However, the usual linear response TDDFT (LR-TDDFT) calculations have some disadvantages for the study of excited-state mechanisms.[73] The most serious problem is that TDDFT provides a discontinuous PES at the crossing point between the reference ground state and the first excited state (S_1), as the reference state becomes the excited state following an intersystem crossing. In dynamics simulations of ultrafast photo-decay processes in which CI regions are very important, LR-TDDFT is not a good choice. Furthermore, LR-TDDFT within the adiabatic approximation cannot describe configurations corresponding to doubly excited states. This limitation can be important for rotations about double bonds, because the ground state and the doubly excited state can cross during such rotations.

To overcome the aforementioned drawbacks to LR-TDDFT, spin-flip (SF) TDDFT was developed.[47,71,74-82] In SF-TDDFT, the lowest energy high spin triplet state is used as the reference state, and both the ground state (S_0) and S_1 are treated as response states following a single electron spin flip. Therefore, the reference state does not change at S_0/S_1 crossing points, and the PES is continuous in the crossing region. Also, the HOMO-LUMO doubly excited state can be described by SF-TDDFT as discussed below. Shao et al. developed the SF-TDDFT analytic gradient.[74] Minezawa and Gordon implemented the SF-TDDFT analytic gradient in the GAMESS program package[76] so it is possible to optimize the minimum energy S_0/S_1-CI SF-TDDFT geometries.[47,76] Also, analytic derivative couplings at the SF-TDDFT level have been developed.[83] Very recently, an automated search for minimum energy SF-TDDFT S_0/S_1-CI geometries has been reported.[81]

In the present study, the interface of the SF-TDDFT method with excited state dynamics simulations is described. Several problems must be addressed. The primary issue is the description of states by the SF-TDDFT method. Since the SF-TDDFT solutions are not spin eigenstates, multiple undesired states appear in the solutions with significant spin contamination. Additionally, identification of the target state during dynamics simulations is complex since state ordering and orbital switching can both occur. To date, no method has been reported that successfully tracks the target state during SF-TDDFT dynamics simulations. In this work, these issues are discussed, and the crossing regions among the SF-TDDFT states are also carefully examined.

Methodology

The SF-TDDFT method can describe all of the excited states that are expressed as linear combinations of Slater determinants obtained by one-electron transitions from the reference state. The spin-flip methodology also successfully exploited in wave-function methods.[85-88] As shown in *Figure 2(a)*, the reference SF-TDDFT state is the high spin triplet state. Response states are generated by the spin-flip of an electron from α to β. Thus, SF-TDDFT can only treat four states correctly; *i.e.* the ground state, the HOMO-LUMO doubly excited state, and the open shell singlet and triplet states of the singly-occupied molecular orbitals (SOMOs), as shown in *Figures 2(b), 2(c)* and *2(d),* respectively. The other states cannot be described correctly in the SF-TDDFT framework, because several of the required determinants cannot be generated by a single spin flip. The missing determinants are indicated by gray crosses in *Figure 2(e).* These unphysical states appear as mixed states of singlet, triplet and quintet spins.

To accomplish SF-TDDFT dynamics simulations, a method is needed that can follow a target state throughout a wide swath of the PES. A trajectory might pass through a crossing region between the target state and some undesired state, such as a mixed state (*Figure 2(e)*)or a triplet state. These undesired states should be ignored in crossing regions, but unequivocal identification of the target state at each dynamics step is not straightforward. It was reported that $<\mathbf{S}^2>$ values for singlet states are commonly 0.0 - 0.2 at local minima or minimum energy CI coordinates,[79] while $<S^2>$ for the triplet state and the mixed states is normally ~2.0 and ~1.0, respectively. However, due to the SF-TDDFT spin contamination problem noted above, $<S^2>$ can deviate considerably from the exact values and can vary significantly during a simulation. Therefore, it is not feasible to identify the target states by monitoring only $<S^2>$ values. Although the use of approximate spin projection formulas may be useful for this problem,[89-92] they cannot easily be applied, because of the possible existence of many undesired states. Previous authors have discussed this issue in some detail.[79,81] Various spin adaptation techniques have been considered as a solution to the spin-contamination issues of SF-TDDFT.[78,87,88,93] For the purpose of the present study, a method to distinguish the target state automatically based on properties other than $<S^2>$ values is considered.

Figure 3a schematically illustrates the *state tracking* method that follows the target state during the SF-TDDFT dynamics simulations. In *Figure 3*, black circles indicate the target state, and the black triangles indicate the SF-TDDFT undesired states. The present method monitors several types of data at every point along trajectories: the energies of the states, the eigenvectors of the states (the CI coefficients), the ordering of the MOs, and the $<S^2>$ values of the states. The first criterion of the *state tracking* method is that the target state is selected as long as each value

is continuous between the previous step and the present step. However, the monitored values sometimes do not provide meaningful information especially in the crossing region as discussed below, and the state tracking sometimes fails.

The ordering of the MOs is maintained at each step by using a built-in GAMESS option. By fixing the ordering of the MOs, the ordering of the eigenvectors is also maintained, and the comparison from step to step becomes much easier. One possible approach is to calculate the dot product of the eigenvectors of the states. However, this strategy sometimes does not work well, because the signs of the elements in the eigenvectors of the states frequently change. Thus, such a method can misread the target state. This sign change is caused by the solution of the LR SF-TDDFT equations and is difficult to control. Thus, two additional eigenvector properties are monitored, as shown in Eqs (1) and (2) below. The CI coefficients of two successive eigenvectors, n and n+1, are represented by the terms labeled c_i.

$$V_a = \sum_i \left| c_i^n c_i^{n+1} \right| \tag{1}$$

$$V_d = \sum_i \left\| c_i^n \right| - \left| c_i^{n+1} \right\| \tag{2}$$

The target state is chosen as the state that has the largest V_a and smallest V_d values in the present method. By using V_a and V_d, the method can almost always identify the target state correctly.

Figure 3(b) shows a typical SF-TDDFT crossing region of the PES. The nature of the two crossing states always changes in the crossing region (indicated by the white squares in *Figure 3(b)*), and the crossing PESs resemble an avoided crossing. Consequently, it is very difficult in

the crossing region to distinguish the target state based only on the eigenvectors and the $<S^2>$ value. Likewise, it is difficult to distinguish the target state based on the energies and the gradient vector due to the nature of the avoided crossing. In the present scheme, when the target state is well isolated from the adjacent states in energy, the eigenvectors of the previous step are compared with those of the present step. In mixing regions, the last step of the previous stable region (*k* in *Figure 3(b)*) and the present step are compared. After the trajectory passes through the mixing region, the method updates the reference eigenvectors. This strategy works well in most SF-TDDFT dynamics simulations.

In the present study, non-adiabatic transitions are not considered, because the calculation of non-adiabatic coupling matrix elements (NACME) within the SF-TDDFT formalism is not yet available in GAMESS. The energy difference threshold between S_1 and S_0 for the termination of the trajectories is defined as 0.1eV or less. Due to the current lack of non-adiabatic transition probabilities, the lifetime of the excited state and the quantum yield of the photoisomerization cannot be discussed quantitatively. However, the qualitative time relationship between the experimental decay time and the statistical time in which the trajectories reach the crossing region from the Franck-Condon (FC) region are summarized, and the mechanism of the photoreaction branching in the excited state is discussed.

The present method does not work well in two situations. First, consider a case in which the trajectory remains in a mixing region for more than ~10 fs. Then, the comparison of the eigenvectors becomes meaningless, since the molecular coordinates are very different from those of the last step in the stable region. Although the trajectories pass through the crossing region in

less than 1 fs in most cases in the present study, the occurrence of trajectories lingering in the crossing region is not negligible due to the large number of crossings along dynamics simulations. Second, difficulties occur when the absolute values of the molecular orbital coefficients for the open shell triplet and the open shell singlet SOMOs are too similar. Since the elements of V_a and V_d are likely to be very similar in this situation, it is difficult to distinguish these states. To improve the treatment for the latter situation, the two values are scaled by the energy difference between the calculated state and the target state, and only states within ±1 eV from the target state are considered in the present *state tracking* method.

The main purpose of the present study is to clarify the photoreaction branching mechanism of *cis*-stilbene using the SF-TDDFT dynamics simulation method. An advantage of the proposed dynamics method is that relatively accurate and low-cost excited state dynamics simulations become possible with SF-TDDFT, while also avoiding the need to choose active spaces.

Computational Details

The dependence of SF-TDDFT predictions on the chosen functional and basis set was analyzed by performing single point energy calculations at several previously reported stilbene geometries.[47] Based on benchmarking of a variety of density functionals on stilbene and basis sets, the BHHLYP functional with the 6-31G(d) basis set was chosen for all of the stilbene excited state dynamics simulations and reaction path calculations. The SF-TDDFT *cis*-stilbene vertical excitation energies are compared with both experiment and *ab initio* theoretical

predictions reported in previous studies.[37,45,47,50] The vertical excitation energy was also calculated with CASPT2(2,2)/cc-pVDZ in order to isolate the effect of dynamic correlation. The (2,2) active space includes the HOMO(π) and the LUMO(π^*). The ground state calculations, including equilibration, were performed at the DFT(BHHLYP)/6-31G(d) level.

To understand the dominant decay pathway after the vertical excitation of *cis*-stilbene, a SF-TDDFT(BHHLYP)/6-31G(d) steepest descent path calculation in the $\pi\pi^*$ state was performed, starting from the Franck-Condon (FC) structure of *cis*-stilbene. A minimum energy conical intersection (MECI) was located by using the penalty function method[76] with SF-TDDFT(BHHLYP)/6-31G(d). The description of conical intersections by several electronic structure methods has been discussed extensively, and the SF-TDDFT method was shown to give the correct shapes of conical intersections.[84] Also, a transition state (TS) and corresponding intrinsic reaction coordinate (IRC) path in the $\pi\pi^*$ state have been determined with SF-TDDFT(BHHLYP)/6-31G(d).

In order to sample initial conditions for the excited state dynamics simulations, an equilibration dynamics simulation in the ground state of *cis*-stilbene was performed for 5 ps, followed by a 20 ps production run. In the equilibration dynamics simulation, the bath temperature was set to 300 K by a Nose-Hoover thermostat.[94] The initial coordinates and velocities for the excited state dynamics were taken once every 400 fs from the equilibration dynamics simulation in the ground state; i.e. 50 trajectories were calculated. The excited state dynamics simulations were started upon excitation to the lowest $\pi\pi^*$ singlet state. Each trajectory in the excited state was terminated if the trajectory reached the S_1/S_0-crossing region, or if the

simulation time reached 1.5 ps (with respect to a 1.2 ps experimental lifetime) in order to manage the computational cost. Recall that the statistical treatment of non-adiabatic transitions is beyond the scope of this work. All of the DFT and MP2 calculations were performed with the GAMESS program package,[95,96] while the CASPT2 calculations were performed with the MOLPRO2010 program package.[97]

Results and Discussion

In *Table 1*, the lowest vertical excitation energies of cis-stilbene obtained by both experimental and theoretical studies are summarized. The experimental vertical excitation energy varies from 4.0 eV to 4.7 eV,[8,16,18,20,23,29,30,33-35] but based on the most recent experiments,[23,29,30,33-35] the vertical excitation energy of cis-stilbene is ~4.6 eV. CASSCF generally predicts excitation energies that are more than 1 eV higher than experimental results. The addition of dynamic correlation via CASPT2 provides reasonable agreement with experiment for the relatively small active spaces. The same is true for SF-TDDFT, as long as a hybrid functional is used. If one uses GGA functionals, the agreement is much worse as shown in *Table 1*. The failure of GGA functionals for excited states has been noted in a recent benchmarking survey.[98] It is concluded that dynamic correlation is important to accurately describe the $\pi\pi^*$ state of stilbene.

To examine the SF-TDDFT basis set dependence, single point energies at several stilbene geometries were calculated with the BHHLYP functional and a variety of basis sets. In *Table 2*, all of the $(S_0)_{trans\text{-min}}$ reported geometries were optimized using the indicated basis set; the

remaining single point energies in the table were calculated at the geometries reported by Minezawa *et al*.[47] As shown in *Table 2*, the SF-TDDFT basis set dependence is very small. This small basis set dependence is an advantage of SF-TDDFT, because the computational cost can be lowered by choosing a relatively small basis set without a significant loss of accuracy.

In *Figure 4*, the definitions of the structural parameters used in the present study are summarized. Five structural parameters are defined to describe the photoisomerization mechanism of *cis*-stilbene, *i.e.* r_{C1C2}, r_{C1C3}, r_{C13C14}, a_{C1C2C4} and $d_{C3C1C2C4}$. r_{ij} indicates the distance between two atoms, a_{ijk} indicates the angle that connects three atoms *ijk*, and d_{ijkl} indicates the dihedral angle between the two planes formed by atoms *ijk* and *jkl*. Each structural parameter of the optimized geometries for $(S_0)_{cis\text{-min}}$ is summarized in *Table 3*. The experimental values measured by X-ray analysis are also shown. The optimized geometries obtained with each method in *Table 3* show good agreement with the experimental values.[99] This led to the consistent choice of DFT(BHHLYP)/6-31G(d) for the equilibration dynamics simulation in the ground state of *cis*-stilbene.

In the present study, two different S_1/S_0-crossing regions were found during the dynamics simulations. Two twisted-pyramidalized MECI structures, $(S_1/S_0)_{twist\text{-I}}$ and $(S_1/S_0)_{twist\text{-II}}$, were located. As shown in *Figure 5*, $(S_1/S_0)_{twist\text{-II}}$ has a larger $d_{C3C1C2C4}$ dihedral angle than $(S_1/S_0)_{twist\text{-I}}$. $(S_1/S_0)_{twist\text{-I}}$ and $(S_1/S_0)_{twist\text{-II}}$ correspond to the MECI structures previously reported in refs 47,50 and refs 45,49, respectively. *Figure 6(a)* shows the variations of structural parameters along the SF-TDDFT(BHHLYP)/6-31G(d) steepest descent paths from the FC structure of *cis*-stilbene in the ππ* state (denoted $(S_1)_{FC}$). $(S_1)_{FC}$ and $(S_0)_{cis\text{-min}}$ indicate the same geometries, but different

electronic states. r_{C13C14} and $d_{C3C1C2C4}$ were plotted to develop an understanding of the photoreaction mechanism of *cis*-stilbene in the ππ* state. The minima in the ground state are indicated as black circles. DHP in the ground state $(S_0)_{DHP\text{-}min}$ has a short r_{C13C14} distance and a small $d_{C3C1C2C4}$ angle, while *trans*-stilbene in the ground state $(S_0)_{trans\text{-}min}$ has a long r_{C13C14} and a large $d_{C3C1C2C4}$ angle. Therefore, these two parameters that correspond to the axes in *Figure 6(a)* can efficiently differentiate the photoreaction branching mechanisms from *cis*-stilbene to both *trans*-stilbene and DHP. The same plots are used to discuss the trajectories of the dynamics simulations in *Figures 7(a)-7(d)*.

As shown in *Figure 6(a)*, the steepest descent path from $(S_1)_{FC}$ leads to a *cis*-stilbene-like structure (referred to as $(S_1)_{cis\text{-}min}$) in the first stage, where movements of two hydrogen atoms are dominant, resulting in a geometry consisting of two near-planar structures of H-C-C_6H_5.[31] This *cis*-stilbene-like structure is reported to be a true minimum on the S_1-PES in several studies,[31,37-39,45,50] although some studies indicate otherwise.[36,47,50] At the present SF-TDDFT(BHHLYP)/6-31G(d) level of theory, the steepest descent path goes through a very-flat region of the PES in the range $s = 4.0 \sim 5.0$ bohr amu$^{1/2}$ around $(S_1)_{cis\text{-}min}$, with a large reaction-path curvature as shown in *Figure 6(b),* and eventually reaches $(S_1)_{DHP\text{-}min}$. This result is interesting because previous studies predict that the reaction pathway leading from the FC region of *cis*-stilbene to the twisted structures is preferred.[5,7,13] It is suggested that branching for the twisted structure and the DHP structure occurs in this sharply curved region, and that dynamic effects may be important in determining the branching ratio of the products. It is also noted that the molecule should stay at this very-flat region around $(S_1)_{cis\text{-}min}$ for some length of time. Although $(S_1)_{cis\text{-}min}$ does not correspond to a true minimum, it sits is a very flat region of the PES.

To clarify the photoreaction dynamics of *cis*-stilbene in the ππ* state, 50 trajectories were calculated using SF-TDDFT(BHHLYP)/6-31G(d). Trajectories going toward both the DHP and twist directions are sampled by the dynamics simulations. 64% of the trajectories reach the S_1/S_0-crossing region near $(S_1/S_0)_{DHP}$, $(S_1/S_0)_{twist\text{-}I}$ or $(S_1/S_0)_{twist\text{-}II}$ within t = 1.5 ps, while the remaining trajectories linger on the S_1-PES. Based on the r_{C13C14} and $d_{C3C1C2C4}$ values at the terminal points of the respective trajectories, the 50 trajectories are divided into three groups: 13 trajectories (26%) lead to the formation of DHP, 35 trajectories (70%) undergo rotation *via* the torsion of the C1=C2 bond, and two trajectories (4%) first began torsional rotation, and then change to DHP. As mentioned previously, quantum yields of the photoreaction cannot be discussed from the present dynamics simulations in a rigorous way, because the trajectories branch into three structures, *i.e.* DHP, *cis*-stilbene and *trans*-stilbene, after relaxing to the ground state. However, the calculated branching ratio (*trans*-stilbene : DHP = 35 : 13) is in good agreement with experimental data (*trans*-stilbene : DHP = 35 : 10), and indicates that the dynamics simulations qualitatively reproduce the experimental quantum yield.

Figures 7(a), 7(b), and *7(c)* show examples of typical trajectories which pass through each of three S_1/S_0-crossing regions. The structural parameters along each trajectory and the S_1/S_0-crossing geometries that the trajectories pass through are also shown. The blue squares indicate the terminal points where the energy differences between S_1 and S_0 becomes 0.1 eV or less. It should be noted that the terminal points of the respective trajectories are not exactly the same as the S_1/S_0-MECI points; the S_1/S_0-crossing regions are distributed around the S_1/S_0-MECI points in configuration space. All of the trajectories from the FC region of *cis*-stilbene move in the direction of $(S_1)_{cis\text{-}min}$ in the early stage of the photo-decay process, and all but two of the

trajectories remain on a single path toward either DHP or one of the two twisted MECIs. Thus, it is expected that DHP and the twisted region in the ππ* state are separated by a barrier between $(S_1)_{cis\text{-min}}$ and $(S_1)_{twist\text{-min}}$. Although the trajectories were not terminated by following the termination criteria described above, the S_1/S_0 energy differences of the two trajectories decreased to < 0.15 eV one time.

The barrier height on the SF-TDDFT(BHHLYP)/6-31G(d) IRC path from $(S_1)_{DHP\text{-min}}$ to $(S_1)_{twist\text{-min}}$ *via* $(S_1)_{TS}$ was calculated to be only 0.1 eV, as shown in *Figure 6(a)*. This TS was also reported in a previous study.[47,50] As indicated in *Figure 6(c)*, the energy variation along the IRC path is very flat around the region s = -12 ~ -2 bohr amu$^{1/2}$, which should include the $(S_1)_{cis\text{-min}}$ structure. Thus, it is expected that the photoreaction branching in the ππ* state of *cis*-stilbene occurs in $(S_1)_{cis\text{-min}}$ where the steepest descent path from $(S_1)_{FC}$ curves dramatically and the PES is very flat.

The calculated population decay of *cis*-stilbene in the ππ* state is shown as a black line in *Figure 8*. To calculate the population of the ππ* state, the trajectories terminated by reaching S_1/S_0-crossing regions are regarded as the trajectories decaying from S_1 to S_0. 28 trajectories reached the S_1/S_0-crossing region before 1.0 ps. As a consequence, as shown in *Figure 8*, the population of the ππ* state at t = 0 is 1.0, *i.e.* all the trajectories are in the ππ* state, and the population decreases to 0.44 at 1.0 ps. As mentioned before, the lifetimes cannot be discussed rigorously, because non-adiabatic coupling calculations were not performed. However, the time scale of the calculated population decay for the ππ* state is in good agreement with the experimental decay of 1.2 ps.[35] The calculated population decay of the ππ* state is also plotted

for the trajectories leading to the *twist*-side (35 trajectories) and for the trajectories leading to DHP-side (13 trajectories), separately, in *Figure 8*. It is clearly shown that the lifetime for the DHP-side is relatively longer than that for the twist-side. This difference can be understood by considering the initial atomic motions for $\pi\pi^*$ excited *cis*-stilbene. Since the steepest descent path starting at $(S_1)_{FC}$ denotes a direction with increasing torsional angle, $d_{C3C1C2C4}$, the torsional motion will receive kinetic energy in the early stage. The direction is related to a structural transformation toward the twist side, while it is almost perpendicular in the direction leading to the DHP-side, indicating that the molecule should reach the $(S_1/S_0)_{twist}$ region faster than the $(S_1/S_0)_{DHP}$ region. As shown in *Figure 7(c),* the trajectory exhibits strong fluctuations and stays around $(S_1)_{cis\text{-min}}$ for a relatively long time before reaching the $(S_1/S_0)_{DHP}$ region, while the trajectories in *Figure 7(a)* and *7(b)* reach the $(S_1/S_0)_{twist}$ region more smoothly.

As mentioned previously, evidence from femotosecond time-resolved fluorescence spectra led to the proposal that the photoreaction process of *cis*-stilbene is a two-step mechanism.[35] Experimental results include measurements of the oscillator strengths for the initial 0.23 ps fast step (0.32), and that of the 1.2 ps second step that follows (0.21).[35] Based on the energy variations of the S_0 and S_1 states along the IRC in *Figure 6(c),* fluorescence can only be observed when the molecule stays around the $(S_1)_{cis\text{-min}}$ region. This is because the observed fluorescence wavelength, 420 nm,[35] nearly coincides with the energy difference between the S_0 and S_1 states, 3.1 eV, for $(S_1)_{cis\text{-min}}$ in *Figure 6(c)*, while the energy difference between the S_0 and S_1 states in other regions along the IRC is too small to be observed. Thus, it is suggested that both the fast step and the second fluorescence decay in the experiment[35] indicate an escape of the molecule from the $(S_1)_{cis\text{-min}}$ region. Therefore, the fast decay may correspond to an escape of the

molecules going to the twist-side, while the second decay may correspond to an escape of the molecules headed to the DHP-side. This attribution is consistent with the decay times of the twist and DHP sides shown in *Figure 8*. It is also interesting that the larger oscillator strength of the fast decay mode corresponds to the major product, *i.e.* twist-side, while the second decay with smaller oscillator strength corresponds to the DHP-side.

Finally, consider the photoreaction branching mechanism of *cis*-stilbene in the ππ* state. As mentioned above, there are three types of related results: the experimental quantum yields of the photoreaction,[5,7,13] the steepest descent path, and the trajectories of the dynamics simulation. From the experimental results, the *cis-trans* isomerization is preferred to the *cis*-DHP cyclization in the ππ* state, although the steepest descent path connects $(S_1)_{FC}$ and $(S_1)_{DHP\text{-}min}$ directly. Also, since the *cis-trans* photoisomerization proceeds more favorably than the *cis*-DHP photocyclization in dynamics simulations, dynamic effects are important as mentioned above.

The branching mechanism in the ππ* state can be explained based on the nature of the PES around $(S_1)_{cis\text{-}min}$-$(S_1)_{TS}$-$(S_1)_{twist\text{-}min}$ and around $(S_1)_{FC}$. As mentioned above, since the energy barrier between $(S_1)_{cis\text{-}min}$ and $(S_1)_{TS}$ is only 0.1 eV even though the path-length is relatively long, the PES is very flat in the region of $(S_1)_{cis\text{-}min}$-$(S_1)_{TS}$-$(S_1)_{twist\text{-}min}$. Thus, it is easy to pass through the barrier between DHP- and the twist-side. In the FC region of *cis*-stilbene, the steepest descent direction promotes a slight increase of the torsional angle, $d_{C3C1C2C4}$. The torsional motion receives kinetic energy in the early stage of the photo-decay process of *cis*-stilbene. If the kinetic energy corresponding to the torsional motion is not sufficient to overcome the barrier between $(S_1)_{cis\text{-}min}$ and $(S_1)_{twist\text{-}min}$, the trajectory will follow the steepest descent path and may lead to

$(S_1/S_0)_{DHP}$. On the other hand, if the molecule can overcome the barrier between $(S_1)_{cis\text{-min}}$ and $(S_1)_{twist\text{-min}}$, then it should reach $(S_1/S_0)_{twist\text{-I}}$ or $(S_1/S_0)_{twist\text{-II}}$. This means that the molecular motion in the early stages following photoexcitation determine the photoreaction branching of *cis*-stilbene. Also, it is clear that the low barrier between $(S_1)_{cis\text{-min}}$ and $(S_1)_{twist\text{-min}}$ plays an important role in the branching ratio of the photoreaction of *cis*-stilbene. This is the first report that discusses the full photoreaction mechanism for *cis*-stilbene, involving both *cis*-DHP cyclization and *cis*-*trans* isomerization based on on-the-fly excited state dynamics simulations.

Conclusions

This is the first attempt to examine the photoreaction process of *cis*-stilbene by full-dimensional on-the-fly dynamics simulations, in order to examine the photoreaction branching mechanism of *cis*-stilbene in the ππ* state. In SF-TDDFT calculations, the characters of nearly degenerate electronic states mix with each other, which makes it difficult to follow a target state along a trajectory in a simple way. To solve this problem, a state tracking method is newly proposed to follow the target state along trajectories, in which reference eigenvectors are used to distinguish states from one another. In the *cis*-stilbene ππ* state the steepest descent path is shown to lead to $(S_1)_{DHP}$ directly from $(S_1)_{FC}$, although the *cis*-*trans* isomerization is preferred to the *cis*-DHP cyclization in experiments. Although non-adiabatic coupling calculations were not performed, the branching ratio calculated from dynamics simulations (*trans* : DHP = 35 : 13) qualitatively reproduces the experimental quantum yield very well. This result indicates that dynamic effects play a significant role in the photoreaction branching mechanism of *cis*-stilbene. The hopping time from the ππ* state to the ground state in dynamics simulations is in good

agreement with the decay time of the femotosecond time-resolved fluorescence spectra (~1.2 ps).[35]

The branching mechanism for ππ*-excited *cis*-stilbene is analyzed in a two-dimensional coordinate space of r_{C13C14} (related to *cis*-DHP cyclization) and $d_{C4C2C1C3}$ (related to *cis*-*trans* isomerization). Three relevant MECIs have been located between the ground and ππ* states of *cis*-stilbene including a new MECI $(S_1/S_0)_{twist\text{-II}}$. Directly following photoexcitation, trajectories starting from the FC region of *cis*-stilbene go downhill in the direction of $(S_1)_{cis\text{-min}}$, and then bifurcate toward DHP or twisted geometries. It is shown that 64% of the trajectories reach the S_1/S_0-crossing region near $(S_1/S_0)_{DHP}$, $(S_1/S_0)_{twist\text{-I}}$ or $(S_1/S_0)_{twist\text{-II}}$ within t = 1.5 ps, while the remaining trajectories linger on the S_1-PES. The exploration of the S_1-PES clarifies that there is a very-low barrier from $(S_1)_{cis\text{-min}}$ to $(S_1)_{twist\text{-min}}$ (~ 0.1 eV), and the PES is very flat around the $(S_1)_{cis\text{-min}}$ region where the branching should occur. The downhill direction at $(S_1)_{FC}$ corresponds to a rotational motion about the central C=C bond, and the barrier height for the path from $(S_1)_{cis\text{-min}}$ to $(S_1)_{twist\text{-min}}$ is very low compared with the kinetic energy of the torsional motion. As a consequence, the *cis*-*trans* photoisomerization is preferred to *cis*-DHP photocyclization in the ππ* state of *cis*-stilbene, and dynamic effects decide the branching rate.

Finally, it should be mentioned that an additional improvement in the state tracking method is required in order to apply SF-TDDFT dynamics simulations to general photoreactions. However, SF-TDDFT is clearly a feasible method for performing excited-state dynamics simulations with a qualitative accuracy and low-cost, and applications of the method to large chemical systems would be of interest.

Acknowledgements

Y. H. acknowledges support from the Japan Society for the Promotion of Science for Research Fellowships for Young Scientists. K. K. and M.S.G. have been supported by a U.S. National Science Foundation Software Infrastructure (SI2) grant, ACI - 1047772. We are sincerely grateful to Prof. Tahei Tahara (RIKEN) and Dr. Satoshi Takeuchi (RIKEN) for valuable discussions and for providing valuable comments on our manuscript.

References

1. Kay, E. R.; Leigh, D. A.; Zerbetto, F. *Angew. Chem. Int. Ed.* **2007**, *46*, 72.
2. Tegeder, P. *J. Phys. Condens. Matter* **2012**, *24*, 394001/1.
3. Szymański, W.; Beierle, J. M.; Kistemaker, H. A. V.; Velema, W. A.; Feringa, B. L. *Chem. Rev.* **2013**, *113*, 6114.
4. Moore, W. M.; Morgan, D. D.; Stermitz, F. R. *J. Am. Chem. Soc.* **1963**, *85*, 829.
5. Muszkat, K. A.; Fischer, E. *J. Chem. Soc. B* **1967**, 662.
6. Saltiel, J. *J. Am. Chem. Soc.* **1968**, *90*, 6394.
7. Wismonskiknittel, T.; Fischer, G.; Fischer, E. *J. Chem. Soc., Perkin Trans.* **1974**, 1930.
8. Greene, B. I.; Farrow, R. C. *J. Chem. Phys.* **1983**, *78*, 3336.
9. Myers, A. B.; Mathies, R. A. *J. Chem. Phys.* **1984**, *81*, 1552.
10. Doany, F. E.; Hochstrasser, R. M.; Greene, B. I.; Millard, R. R. *Chem. Phys. Lett.* **1985**, *118*, 1.
11. Petek, H.; Fujiwara, Y.; Kim, D.; Yoshihara, K. *J. Am. Chem. Soc.* **1988**, *110*, 6269.

12. Abrash, S.; Repinec, S.; Hochstrasser, R. M. *J. Chem. Phys.* **1990**, *93*, 1041.

13. Petek, H.; Yoshihara, K.; Fujiwara, Y.; Zhe, L.; Penn, J. H.; Frederick, J. H. *J. Phys. Chem.* **1990**, *94*, 7539.

14. Todd, D. C.; Jean, J. M.; Rosenthal, S. J.; Ruggiero, A. J.; Yang, D.; Fleming, G. R. *J. Chem. Phys.* **1990**, *93*, 8658.

15. Repinec, S. T.; Sension, R. J.; Szarka, A. Z.; Hochstrasser, R. M. *J. Phys. Chem.* **1991**, *95*, 10380.

16. Rodier, J. M.; Ci, X. P.; Myers, A. B. *Chem. Phys. Lett.* **1991**, *183*, 55.

17. Waldeck, D. H. *Chem. Rev.* **1991**, *91*, 415.

18. Nikowa, L.; Schwarzer, D.; Troe, J.; Schroeder, J. *J. Chem. Phys.* **1992**, *97*, 4827.

19. Pedersen, S.; Banares, L.; Zewail, A. H. *J. Chem. Phys.* **1992**, *97*, 8801.

20. Rice, J. K.; Baronavski, A. P. *J. Phys. Chem.* **1992**, *96*, 3359.

21. Sension, R. J.; Szarka, A. Z.; Hochstrasser, R. M. *J. Chem. Phys.* **1992**, *97*, 5239.

22. Todd, D. C.; Fleming, G. R.; Jean, J. M. *J. Chem. Phys.* **1992**, *97*, 8915.

23. Rodier, J. M.; Myers, A. B. *J. Am. Chem. Soc.* **1993**, *115*, 10791.

24. Sension, R. J.; Repinec, S. T.; Szarka, A. Z.; Hochstrasser, R. M. *J. Chem. Phys.* **1993**, *98*, 6291.

25. Todd, D. C.; Fleming, G. R. *J. Chem. Phys.* **1993**, *98*, 269.

26. Baumert, T.; Frohnmeyer, T.; Kiefer, B.; Niklaus, P.; Strehle, M.; Gerber, G.; Zewail, A. H. *Appl. Phys. B* **2001**, *72*, 105.

27. Fuss, W.; Kosmidis, C.; Schmid, W. E.; Trushin, S. A. *Angew. Chem. Int. Edit.* **2004**, *43*, 4178.

28. Fuss, W.; Kosmidis, C.; Schmid, W. E.; Trushin, S. A. *Chem. Phys. Lett.* **2004**, *385*, 423.

29. Ishii, K.; Takeuchi, S.; Tahara, T. A *Chem. Phys. Lett.* **2004**, *398*, 400.

30. Nakamura, T.; Takeuchi, S.; Suzuki, N.; Tahara, T. *Chem. Phys. Lett.* **2008**, *465*, 212.

31. Takeuchi, S.; Ruhman, S.; Tsuneda, T.; Chiba, M.; Taketsugu, T.; Tahara, T. *Science* **2008**, *322*, 1073.

32. Kovalenko, S. A.; Dobryakov, A. L.; Ioffe, I.; Ernsting, N. P. *Chem. Phys. Lett.* **2010**, *493*, 255.

33. Sajadi, M.; Dobryakov, A. L.; Garbin, E.; Ernsting, N. P.; Kovalenko, S. A. *Chem. Phys. Lett.* **2010**, *489*, 44.

34. Weigel, A.; Ernsting, N. P. *J. Phys. Chem. B* **2010**, *114*, 7879.

35. Nakamura, T.; Takeuchi, S.; Taketsugu, T.; Tahara, T. *Phys. Chem. Chem. Phys.* **2012**, *14*, 6225.

36. Dobryakov, A. L.; Ioffe, I.; Granovsky, A. A.; Ernsting, N. P.; Kovalenko, S. A. *J. Chem. Phys.* **2012**, *137*, 244505/1.

37. Improta, R.; Santoro, F. *J. Phys. Chem. A* **2005**, *109*, 10058-10067.

38. Bearpark, M. J.; Bernardi, F.; Clifford, S.; Olivucci, M.; Robb, M. A.; Vreven, T. *J. Phys. Chem. A* **1997**, *101*, 3841.

39. Berweger, C. D.; van Gunsteren, W. F.; Muller-Plathe, F. *J. Chem. Phys.* **1998**, *108*, 8773.

40. Amatatsu, Y. *Chem. Phys. Lett.* **1999**, *314*, 364.

41. Amatatsu, Y. *J. Mol. Struct. THEOCHEM* **1999**, 311.

42. Molina, V.; Merchan, M.; Roos, B. O. *Spectrochimica Acta. Part A* **1999**, *55*, 433.

43. Dou, Y. S.; Allen, R. E. *Chem. Phys. Lett.* **2003**, *378*, 323.

44. Dou, Y. S.; Allen, R. E. *J. Chem. Phys.* **2003**, *119*, 10658.

45. Quenneville, J.; Martinez, T. J. *J. Phys. Chem. A* **2003**, *107*, 829.

46. Dou, Y. S.; Allen, R. E. *J. Mod. Opt.* **2004**, *51*, 2485.

47. Minezawa, N.; Gordon, M. S. *J. Phys. Chem. A* **2011**, *115*, 7901.

48. Liu, F.; Morokuma, K. *J. Am. Chem. Soc.* **2012**, *134*, 4864.

49. Tomasello, G.; Garavelli, M.; Orlandi, G. *Phys. Chem. Chem. Phys.* **2013**, *15*, 19763.

50. Ioffe, I. N.; Granovsky, A. A. *J. Chem. Theory Comput.* **2013**, *9*, 4973.

51. Ishibashi, Y.; Fujiwara, M.; Umesato, T.; Saito, H.; Kobatake, S.; Irie, M.; Miyasaka, H. *J. Phys. Chem. C* **2011**, *115*, 4265.

52. Irie, M. *Chem. Rev.* **2000**, *100*, 1685.

53. Mahboobi, S.; Dechant, I.; Reindl, H.; Pongratz, H.; Popp, A.; Schollmeyer, D. *J. Heterocyclic Chem.* **2000**, *37*, 307-329.

54. Gordon, M. S.; Chaban, G.; Taketsugu, T. *J. Phys. Chem.* **1996**, *100*, 11512.

55. Sun, L. P.; Song, K. Y.; Hase, W. L. *Science* **2002**, *296*, 875.

56. Ootani, Y.; Satoh, K.; Nakayama, A.; Noro, T.; Taketsugu, T. *J. Chem. Phys.* **2009**, *131*, 194306/1.

57. Bernardi, F.; Olivucci, M.; Robb, M. A. *Chem. Soc. Rev.* **1996**, *25*, 321.

58. Yarkony, D. R. *Accounts Chem. Res.* **1998**, *31*, 511.

59. Schroder, D.; Shaik, S.; Schwarz, H. *Accounts Chem. Res.* **2000**, *33*, 139.

60. Sobolewski, A. L.; Domcke, W.; Dedonder-Lardeux, C.; Jouvet, C. *Phys. Chem. Chem. Phys.* **2002**, *4*, 1093.

61. Poli, R.; Harvey, J. N. *Chem. Soc. Rev.* **2003**, *32*, 1.

62. Levine, B. G.; Martinez, T. J. *Annu. Rev. Phys. Chem.* **2007**, *58*, 613.

63. Nanbu, S.; Ishida, T.; Nakamura, H. C*hem. Sci.* **2010**, *1*, 663.

64. Mori, T.; Kato, S. *Chem. Phys. Lett.* **2009**, *476*, 97.

65. Nakano, H. *J. Chem. Phys.* **1993**, *99*, 7983.

66. Nakano, H. *Chem. Phys. Lett.* **1993**, *207*, 372.

67. Nakano, H.; Hirao, K.; Gordon, M. S. *J. Chem. Phys.* **1998**, *108*, 5660.

68. Lischka, H.; Dallos, M.; Szalay, P. G.; Yarkony, D. R.; Shepard, R. *J. Chem. Phys.* **2004**, *120*, 7322.

69. Dallos, M.; Lischka, H.; Shepard, R.; Yarkony, D. R.; Szalay, P. G. *J. Chem. Phys.* **2004**, *120*, 7330.

70. Runge, E.; Gross, E. K. U. *Phys. Rev. Lett.* **1984**, *52*, 997.

71. Casida, M. E. *In recent advances in density functional methods*; World Scientific: Singapore, **1995**.

72. Burke, K.; Werschnik, J.; Gross, E. K. U. *J. Chem. Phys.* **2005**, *123*, 062206/1.

73. Levine, B. G.; Ko, C.; Quenneville, J.; Martinez, T. J. *Mol. Phys.* **2006**, *104*, 1039.

74. Shao, Y. H.; Head-Gordon, M.; Krylov, A. I. *J. Chem. Phys.* **2003**, *118*, 4807.

75. Wang, F.; Ziegler, T. *J. Chem. Phys.* **2004**, *121*, 12191.

76. Minezawa, N.; Gordon, M. S. *J. Phys. Chem. A* **2009**, *113*, 12749.

77. Huix-Rotllant, M.; Natarajan, B.; Ipatov, A.; Wawire, C. M.; Deutsch, T.; Casida, M. E. *Phys. Chem. Chem. Phys.* **2010**, *12*, 12811.

78. Rinkevicius, Z.; Vahtras, O.; Agren, H. *J. Chem. Phys.* **2010**, *133*, 114104/1.

79. Bernard, Y. A.; Shao, Y.; Krylov, A. I. *J. Chem. Phys.* **2012**, *136*, 204103/1.

80. Li, Z.; Liu, W. *J. Chem. Phys.* **2012**, *136*, 024107/1.

81. Harabuchi, Y.; Maeda, S.; Taketsugu, T.; Minezawa, N.; Morokuma, K. *J. Chem. Theory Comput.* **2013**, *9*, 4116.

82. Isegawa, M.; Truhlar, D. G. *J. Chem. Phys.* **2013**, *138*, 134111/1.

83. Zhang, X.; Herbert, J. M. *J. Chem. Phys.* **2014**, *141*, 064104/1.

84. Gozem, S.; Melaccio, F.; Valentini, A.; Filatov, M.; Huix-Rotllant, M.; Ferré, N.; Frutos, L. M.; Angeli, C.; Krylov, A. I.; Granovsky, A. A.et al. *J. Chem. Theory Comput.* **2014**, *10*, 3074.

85. Krylov, A. I. *Chem. Phys. Lett.* **2001**, *350*, 522.

86. Krylov, A. I. *Chem. Phys. Lett.* **2001**, *338*, 375.

87. Krylov, A. I.; Sherrill, C. D. *J. Chem. Phys.* **2002**, *116*, 3194.

88. Sears, J. S.; Sherrill, C. D.; Krylov, A. I. *J. Chem. Phys.* **2003**, *118*, 9084.

89. Yamaguchi, K.; Tsunekawa, T.; Toyoda, Y.; Fueno, T. *Chem. Phys. Lett.* **1988**, *143*, 371.

90. Shoji, M.; Koizumi, K.; Kitagawa, Y.; Kawakami, T.; Yamanaka, S.; Okumura, M.; Yamaguchi, K. *Chem. Phys. Lett.* **2006**, *432*, 343.

91. Wittbrodt, J. M.; Schlegel, H. B. *J. Chem. Phys.* **1996**, *105*, 6574.

92. Chen, W.; Schlegel, H. B. *J. Chem. Phys.* **1994**, *101*, 5957.

93. Casanova, D.; Head-Gordon, M. *Phys. Chem. Chem. Phys.* **2009**, *11*, 9779.

94. Martyna, G. J.; Klein, M. L.; Tuckerman, M. *J. Chem. Phys.* **1992**, *97*, 2635.

95. Gordon, M. S.; Schmidt, M. W. *In theory and applications of computational chemistry: The first forty years*; Elsevier: Amsterdam, The Netherlands, **2005**.

96. Schmidt, M. W.; Baldridge, K. K.; Boatz, J. A.; Elbert, S. T.; Gordon, M. S.; Jensen, J. H.; Koseki, S.; Matsunaga, N.; Nguyen, K. A.; Su, S. J.; Windus, T. L.; Dupuis, M.; Montgomery, J. A. *J. Comput. Chem.* **1993**, *14*, 1347.

97. Molpro, version 2012.1, a package of ab initio programs. http://www.molpro.net (accessed July, 2013)

98. Leang, S. S.; Zahariev, F.; Gordon, M. S. *J. Chem. Phys.* **2012**, *136*, 104101/1.

99. Traetteberg, M.; Frantsen, E. B. *J. Mol. Struct.* **1975**, *26*, 69.

100. Miller, W.H.; Handy, N.C.; Adams, J.E. *J. Chem. Phys.* **1980**, *72*, 99.

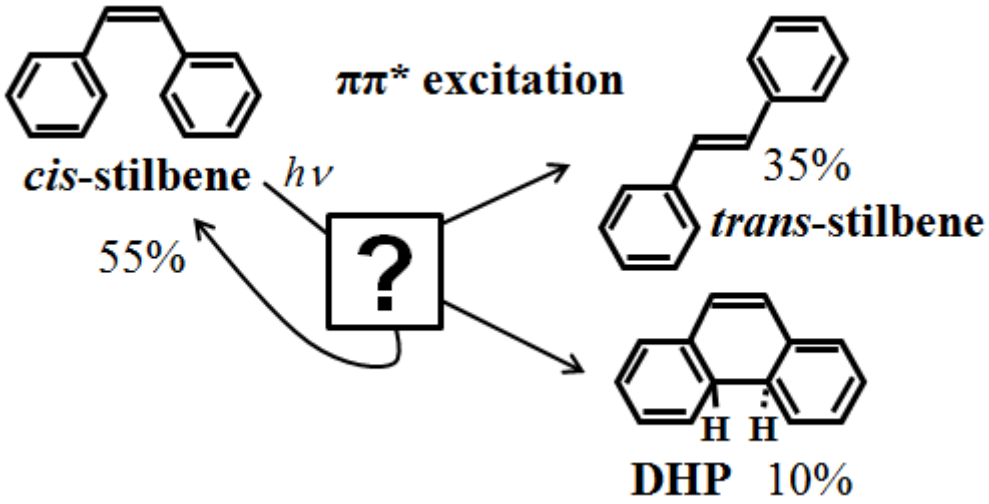


Figure 1. Schematic of the photoisomerization and photocyclization branching of *cis*-stilbene after ππ* excitation. The quantum yields reported in the experimental study[13] are shown as percentages.

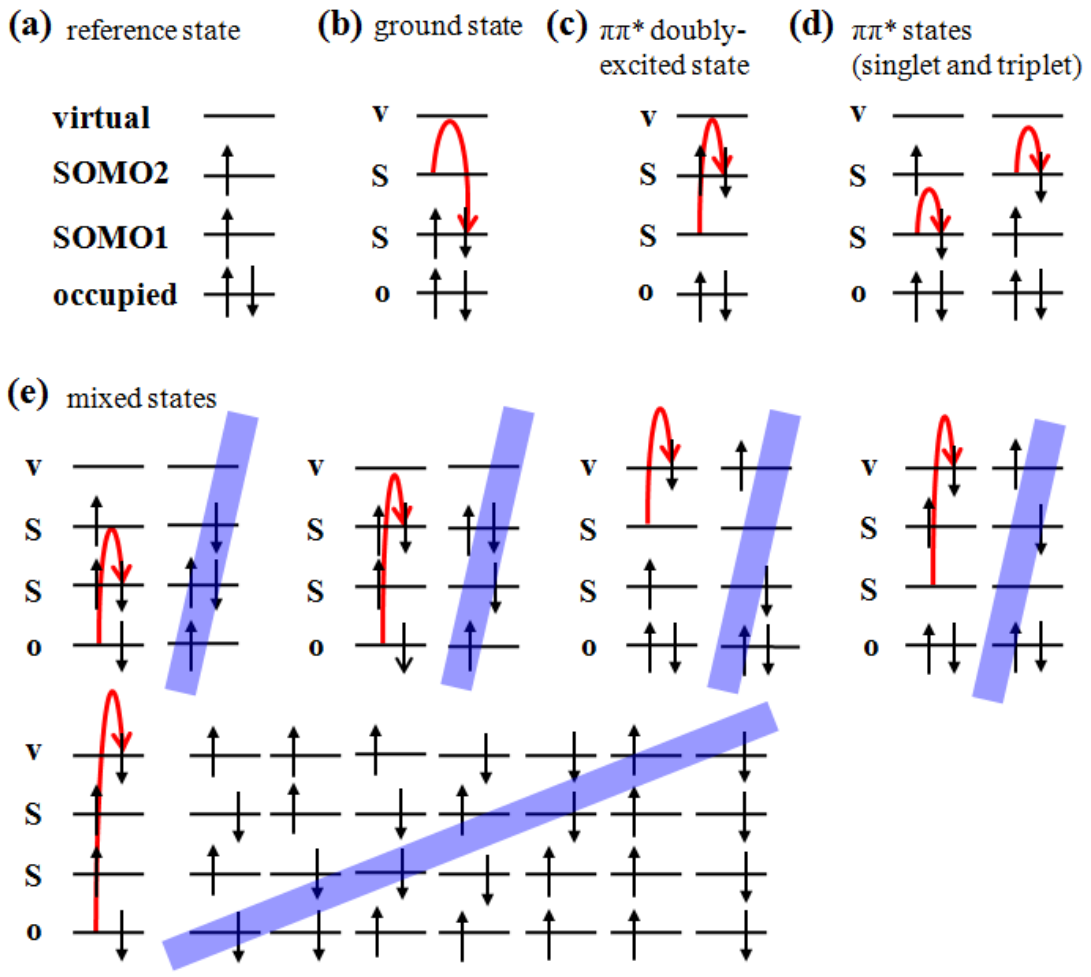


Figure 2. The complete set of configurations considered in SF-TDDFT. (a) depicts the configuration of the reference triplet state. In (b)-(e), all possible Slater determinants obtained by the one-electron spin-flip excitation of the reference triplet state are depicted without blue slashes, and the missing configurations are depicted with blue slashes: (b) denotes the singlet ground state, (c) denotes the [SOMO1(s) → SOMO2(s)] doubly-excited state, (d) denotes the open shell singlet and triplet states of [SOMO1 → SOMO2] and (e) denotes the excited states involving excitations corresponding to the occupied orbitals (o) and the virtual orbitals (v).

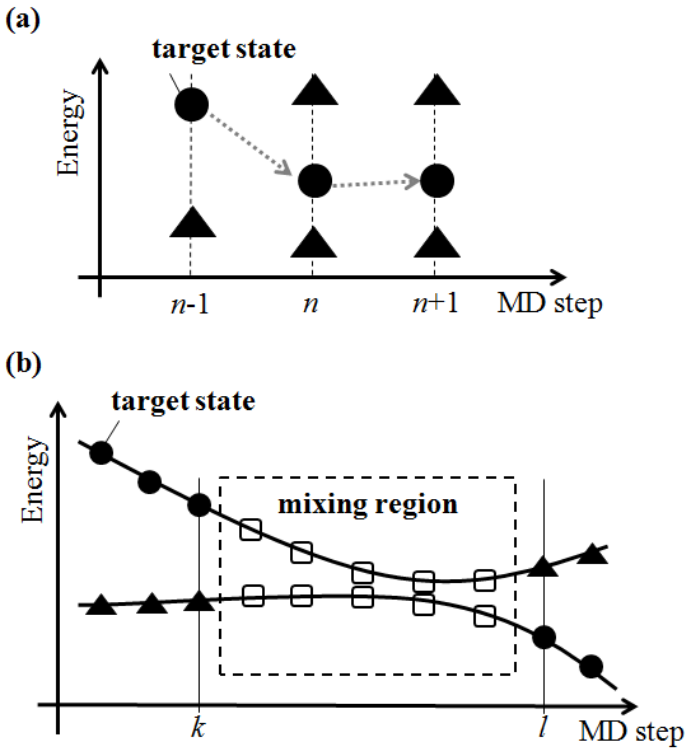


Figure 3. Schematic pictures of a state tracking method. In both pictures, the target states of the dynamics simulation are indicated by black circles, while the undesired states, *i.e.* a triplet state or a mixed state, are indicated by black triangles. (a) illustrates the concept of the state tracking method. The target state is selected from all of the states, as the character of the target state is continuous between the previous step and the present step. (b) depicts a crossing region of the PES along a SF-TDDFT trajectory. The region in which the characters of the two states mix is indicated by dashed lines, and the states in the crossing region in which the target state and an undesired state mix are indicated by white squares. The state tracking method does not update the reference eigenvectors in the mixing region. The state tracking method compares the eigenvector of the last step before the crossing region (indicated by *k*) to the eigenvector in the mixing region, and updates the information after passing through the mixing region (indicated by *l*).

Table 1. Vertical excitation energies of *cis*-stilbene from the ground state.

Level	**ΔE [eV]**
Experiment	4.6[a]
CASSCF(2,2)/6-31G(d,p)	6.07[b]
CASPT2(2,2)/cc-pVDZ	4.23[c]
XMCQDPT2(14,14)/cc-pVDZ	4.43[d]
TDDFT(PBE0)/6-311+G(2d,2p)	4.09[e]
SF-TDDFT(BHHLYP)/DH(d,p)	4.78[f]
SF-TDDFT(B3LYP)/6-31G(d)	3.86[c]
SF-TDDFT(BHHLYP)/6-31G(d)	4.96[c]
SF-TDDFT(PBE0)/6-31G(d)	4.02[c]
SF-TDDFT(BLYP)/6-31G(d)	6.64[c]
SF-TDDFT(BOP)/6-31G(d)	6.66[c]
SF-TDDFT(PBE)/6-31G(d)	6.66[c]

[a]Reference 35. [b]Reference 45. [c]Computed values in the present study. All vertical excitation energies were calculated at the MP2/cc-pVTZ geometry. [d]Reference 50. [e]Reference 37. [f]Reference 47.

Table 2. SF-TDDFT(BHHLYP)/6-31G(d) relative energies (eV) of the ground state and the first excited ππ* states of stilbene. All *trans* isomers are optimized in the ground state with the indicated basis set; the reference energies are the energies of the ground state equilibrium structure of *trans*-stilbene, $(S_0)_{trans\text{-min}}$, for each basis set. The other single point energies were calculated at the geometries reported by Minezawa *et al*.[47] The names of the structures are defined in Ref. 47. Since there is another, twisted-pyramidalized, conical intersection as (*Figure 5(a)*), the reported structure of $(S_1/S_0)_{pyr}$ is denoted $(S_1/S_0)_{twist\text{-I}}$ to distinguish the two MECIs. $(S_1)_{cis\text{-min}}$ is not a true minimum due to the constrained optimization.[44]

	State	6-31G	6-31G(d)	6-31+G(d)	cc-pVDZ	cc-pVTZ	DH(d,p)
$(S_0)_{trans\text{-min}}$	S_0	0.00	0.00	0.00	0.00	0.00	0.00
	S_1	4.70	4.62	4.48	4.53	4.48	4.45
$(S_0)_{cis\text{-min}}$	S_0	0.20	0.19	0.16	0.21	0.24	0.21
	S_1	5.22	5.11	4.92	5.04	4.99	4.99
$(S_0)_{DHP\text{-min}}$	S_0	2.05	1.84	1.80	1.84	1.93	1.75
	S_1	5.17	4.93	4.79	4.88	4.93	4.75
$(S_1)_{twist\text{-min}}$	S_0	3.18	3.11	3.06	3.09	3.15	3.07
	S_1	4.27	4.12	4.00	4.05	4.10	4.05
$(S_1)_{cis\text{-min}}$	S_0	1.05	1.05	1.02	1.06	1.12	1.04
	S_1	4.28	4.19	4.05	4.13	4.13	4.08
$(S_1)_{DHP\text{-min}}$	S_0	2.69	2.64	2.61	2.66	2.74	2.60
	S_1	3.92	3.79	3.73	3.78	3.84	3.72
$(S_1/S_0)_{twist\text{-I}}$	S_0	4.34	4.19	4.13	4.17	4.24	4.16
	S_1	4.52	4.23	4.15	4.18	4.26	4.18
$(S_1/S_0)_{DHP}$	S_0	4.04	3.93	3.89	3.93	4.01	3.87
	S_1	4.13	3.96	3.91	3.95	4.02	3.89

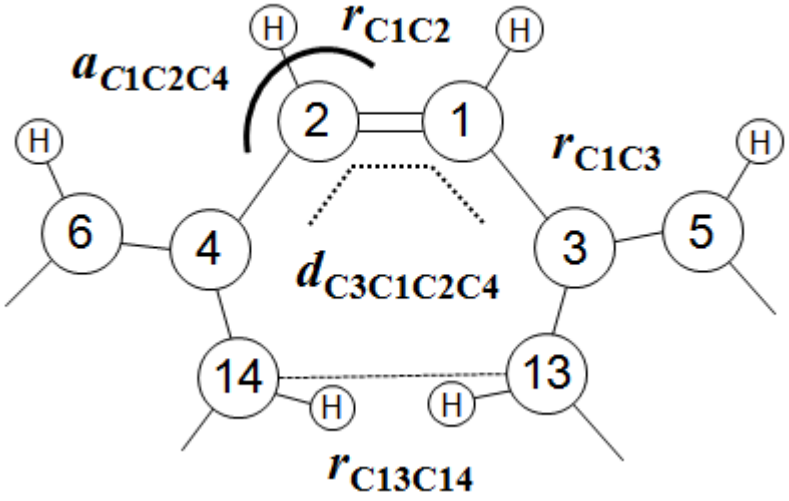


Figure 4. Definitions of the numbering of the atoms and definitions of the internal coordinates. Five structural parameters were defined to discuss the mechanism of *cis*-stilbene, *i.e.* r_{C1C2}, r_{C1C3}, l_{C13C14}, a_{C1C2C4} and $d_{C3C1C2C4}$. r_{C1C2}, r_{C1C3} and r_{C13C14} indicate the distances between two atoms, a_{C1C2C4} indicates the angle connecting three atoms, C1C2C4, and $d_{C3C1C2C4}$ indicates the dihedral angle of two planes, C3C1C2 and C1C2C4.

Table 3. Optimized structural parameters in the ground state of *cis*-stilbene. The three distances, r_{C1C2}, r_{C1C3} and r_{C13C14}, are shown in Å, and the angle and dihedral angle, a_{C1C2C4} and $d_{C3C1C2C4}$, are shown in degrees. The numbering of the atoms is defined in *Figure 4*. The experimental values measured by X-ray structure analysis[99] are also shown.

Method	r_{C1C2}	r_{C1C3}	r_{C13C14}	a_{C1C2C4}	$d_{C3C1C2C4}$
[a]X-ray	1.334	1.489	---	129.5	---
[b]MP2/cc-pVTZ	1.348	1.469	3.127	127.0	5.9
[b]SF-TDDFT(BHHLYP)/6-31G*	1.336	1.472	3.236	129.5	6.3
[b]DFT(BHHLYP)/6-31G*	1.335	1.473	3.249	129.6	6.4

[a]Reference 99. [b]The present study.

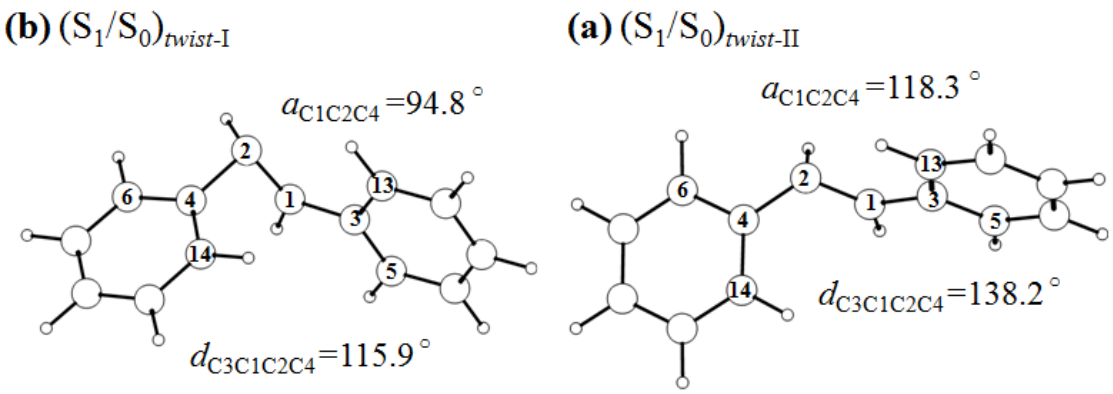


Figure 5. The geometries of two SF-TDDFT S_1/S_0-MECIs. (a) $(S_1/S_0)_{twist\text{-I}}$ was optimized with BHHLYP(SF-TDDFT)/DH(d,p).[47] (b) $(S_1/S_0)_{twist\text{-II}}$ was optimized with BHHLYP(SF-TDDFT)/6-31G(d). $(S_1/S_0)_{twist\text{-II}}$ corresponds to the MECI reported by Quenneville.[45] The angle and the dihedral angle corresponding to the twisted-pyramidalized structures, *i.e.* a_{C4C2C1} and $d_{C3C1C2C4}$ defined in *Figure 4* are also shown.

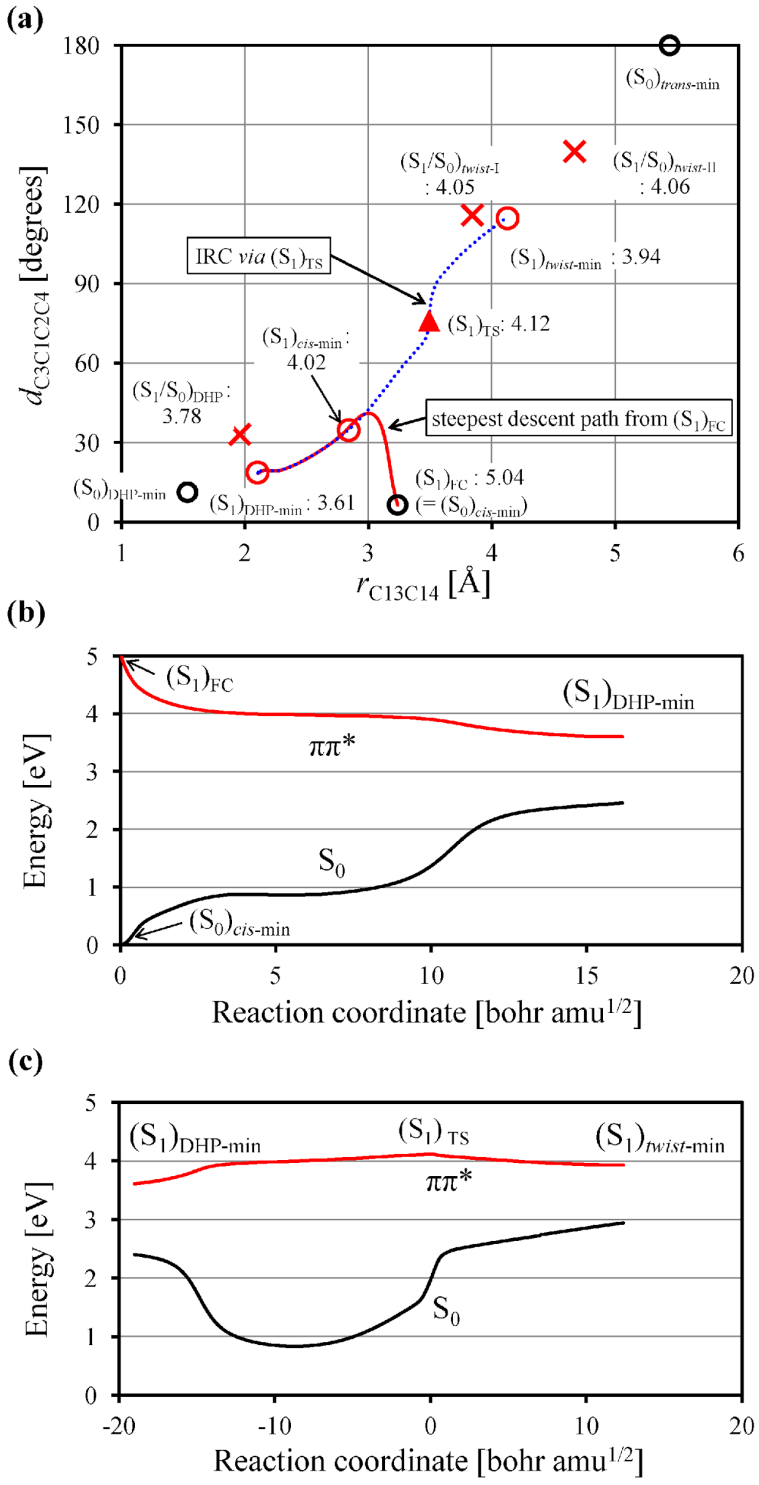


Figure 6. SF-TDDFT(BHHLYP)/6-31G(d) steepest descent path and the IRC path in the ππ* state of stilbene. The reference for the relative energies is the energy of the ground state

equilibrium structure of *cis*-stilbene, $(S_0)_{cis\text{-min}}$. (a) describes the structural parameters in the two-dimensional plot along the steepest descent paths from the Franck-Condon (FC) region of *cis*-stilbene (denoted $(S_0)_{FC}$), and the IRC path *via* the TS between $(S_1)_{DHP\text{-min}}$ and $(S_1)_{twist\text{-min}}$ (denoted $(S_1)_{TS}$). The structural parameters of the minima in the ground state, the minima in the $\pi\pi^*$ state, and the S_1/S_0-MECIs are shown. The x and y axes correspond to r_{C13C14} and $d_{C4C2C1C3}$, respectively, defined in *Figure 4*. The red solid line corresponds to the SF-TDDFT(BHHLYP)/6-31G(d) steepest descent path, and the blue dot line indicates the SF-TDDFT(BHHLYP)/6-31G(d) IRC path *via* $(S_1)_{TS}$. The black circles indicate the minima in the ground state, the red circles indicate minima in the $\pi\pi^*$ state, the red triangle indicates $(S_1)_{TS}$, and the red crosses indicate the S_1/S_0-MECIs. The geometry of $(S_1/S_0)_{twist\text{-II}}$ was optimized with SF-TDDFT(BHHLYP)/6-31G(d) in the present study, and the other MECI and S_1-minimum geometries are calculated with BHHLYP(SF-TDDFT)/DH(d,p).[47] Note that the $(S_1)_{cis\text{-min}}$ is not a true minimum due to the constrained optimization.[47] The relative energies of the indicated geometries are shown in eV. (b) and (c) show the values of the energies along the steepest descent path from $(S_1)_{FC}$ and the IRC path via $(S_1)_{TS}$, respectively. The black solid line indicates the energy of the ground state, and the red solid line indicates the energy of the $\pi\pi^*$ state.

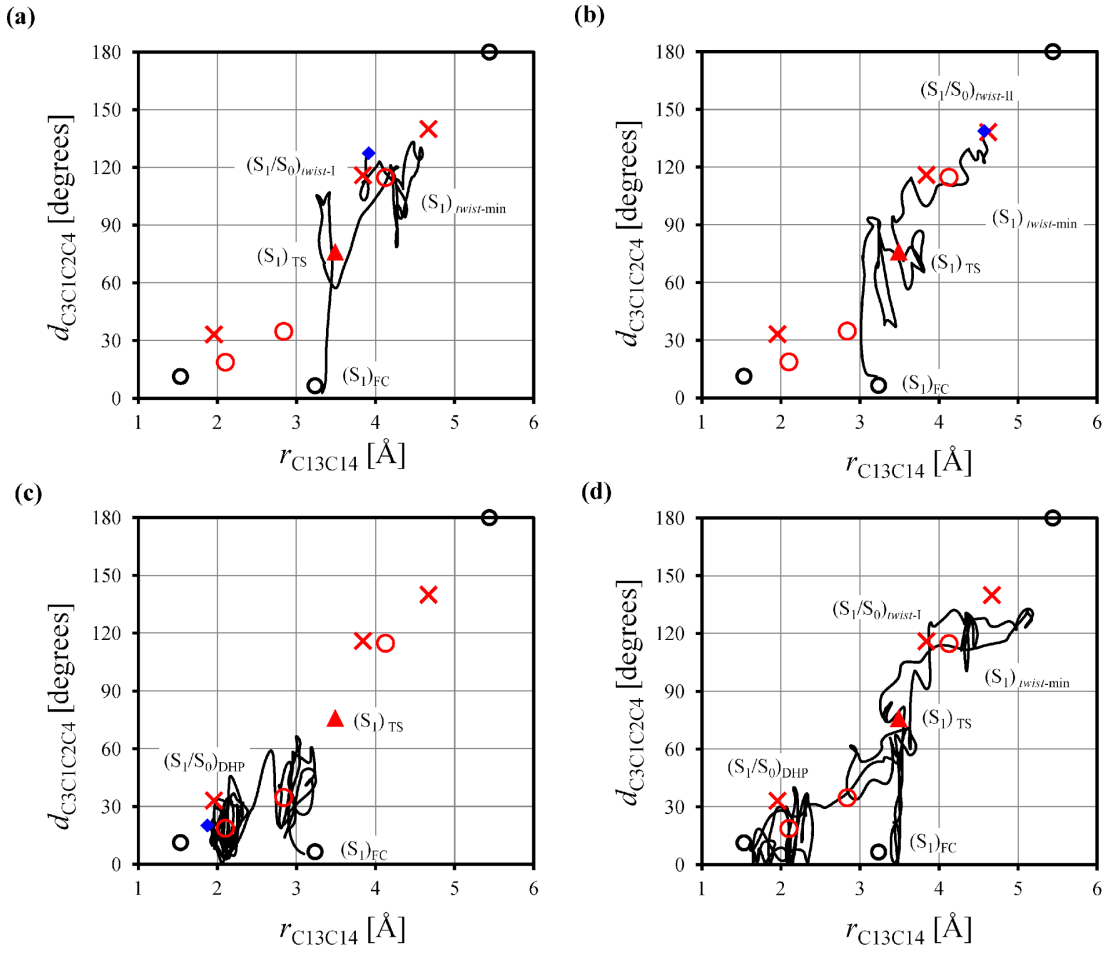


Figure 7. Variations of geometric parameters along three typical types of trajectories [(a), (b), (c)] and a rare trajectory (d), with SF-TDDFT(BHHLYP)/6-31G(d). x and y axes correspond to r_{C13C14} and $d_{C4C2C1C3}$, respectively, defined in *Figure 4*. (a)-(c) indicate the trajectories that reach S_1/S_0-crossing regions corresponding to $(S_1/S_0)_{twist\text{-}I}$, $(S_1/S_0)_{twist\text{-}II}$, and $(S_1/S_0)_{DHP}$, respectively. (d) represents a trajectory that first goes to the twist-side and then crosses over to the DHP-side (this trajectory did not reach the S_1/S_0-crossing until 1.5 ps). Black solid lines indicate the geometric parameters along the trajectories, and blue squares indicate the S_1/S_0-crossing points at which the trajectories were terminated as discussed previously. Black circles indicate minima in the ground state, and red circles indicate minima in the ππ* state; the red triangle indicates $(S_1)_{TS}$, and red crosses indicate the S_1/S_0-MECIs denoted in *Figure 6(a)*.

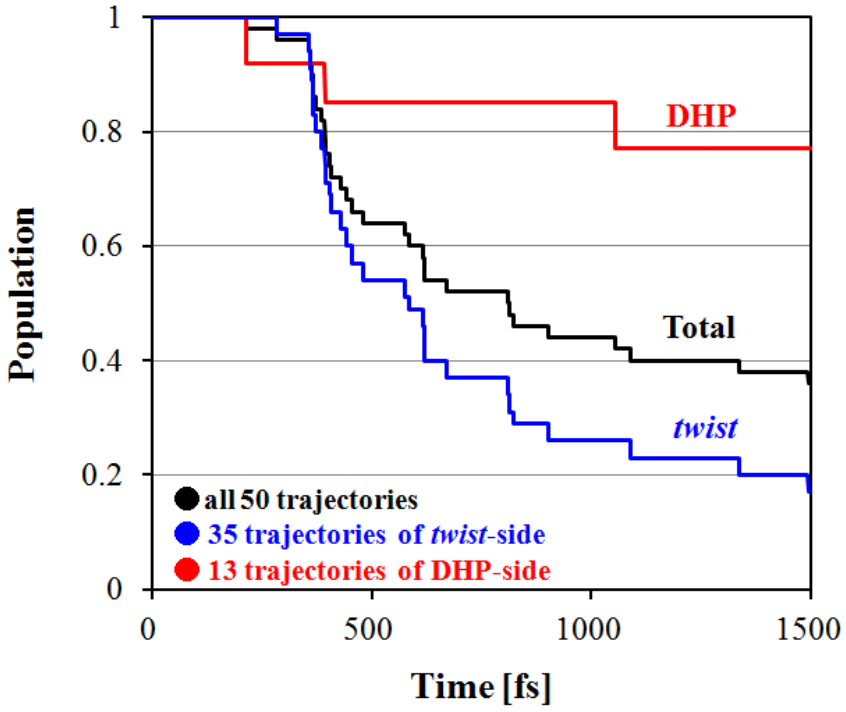


Figure 8. Calculated population decay of the ππ* state for 50 trajectories (black), with the 35 trajectories of the *twist*-side in blue and the 13 trajectories of the DHP-side in red. All trajectories started in the ππ* state at $t = 0$, and the population is 1.0 at $t = 0$.

CHAPTER 8: GENERAL CONCLUSIONS

As we move closer to the exascale era of high performance computing, computer hardware is changing faster than ever before. Computational chemists are working diligently to adapt current methods to next-generation hardware, encouraged by the completely new areas of scientific innovation made possible at exascale. In light of the rapidly evolving computational sciences landscape, the work presented in this dissertation was motivated by three primary topics. First, the viability of novel low-power ARM CPUs for energy-efficient computational chemistry was thoroughly evaluated to reflect the energy consumption challenges associated with exascale computing. Second, the adoption of fundamental quantum chemistry methods to next-generation hardware was discussed from the approaches of both algorithm redesign, and utilization of standalone community software libraries. Third, the current capabilities of computational chemistry were leveraged to investigate applications of heterogeneous catalysis in biodiesel production, and to model photochemical and photoisomerization pathways of *cis*-stilbene.

In Chapter 2, the performance and energy-efficiencies of low-power 32-bit and 64-bit ARM CPUs were compared against commodity workstation-class Intel x86 processors for quantum chemistry workloads. All comparisons were between single-socket CPUs. The Intel processor is the clear choice for minimizing time to solution. Depending on the quantum chemistry method analyzed, between 2 and 4 32-bit or 64-bit ARM cores are required to process a workload as quickly as a single Intel core. In terms of energy to solution, 32-bit ARM is consistently the most efficient processor and 64-bit ARM is consistently the least efficient. The

transition from 32-bit to 64-bit ARM requires additional energy usage, with no observable increase in computational throughput. Furthermore, the idle power consumption of the 18-core Intel CPU is lower than the idle power consumption of the 8-core 64-bit ARM CPU due to low-power CPU states unique to the Intel microarchitecture. While the energy efficiency of 32-bit ARM is promising, it appears that the workstation-class 64-bit ARM CPUs require further development to compete with commodity processors in high performance computing.

In Chapter 3, the performance and energy-efficiency analysis of 64-bit ARM and Intel x86 CPUs was extended to include parallel quantum chemistry methods which utilize connected computer clusters. For a combined metric of time and energy to solution, increasing the number of 64-bit ARM cores used for a computation almost always improves the overall efficiency for the methods analyzed. The only exceptions are jobs which were intentionally configured to heavily utilize hard disk drives for storage of integral data. In general, twice as many 64-bit ARM cores are required to match the time to solution for a given number of Intel x86 cores. Additionally, an in-depth analysis of 64-bit ARM architectural bottlenecks was performed using an extended benchmark set with applications outside of quantum chemistry. It was found that fewer floating point operations are executed per instruction on the 64-bit ARM compared to Intel x86. Furthermore, the 64-bit ARM exhibited inferior memory subsystem performance in terms of main memory and L1 cache read bandwidth.

Implementation details for a new hybrid MPI/OpenMP GAMESS Hartree-Fock algorithm targeting the Intel Xeon Phi Knight's Landing (KNL) processor were presented in Chapter 4. The KNL processor exemplifies the current trend in supercomputing to increase CPU

core counts while reducing the amount of memory per core. By sharing Fock and density matrix data structures which were replicated among computation units in the legacy implementation, the overall memory requirement of the code is reduced by as much as ~200x. The substantial reduction in memory footprint enables utilization of all KNL hardware threads simultaneously, which was demonstrated to reduce the time to solution by 2-3x on average compared to the legacy implementation. Excellent strong scaling was demonstrated while utilizing as many as 3,000 KNL nodes (192,000 cores). This work represents the "algorithm redesign" approach to porting legacy codes to new hardware architectures.

In Chapter 5, challenges in quantum chemistry software interoperability were discussed. Interoperability is useful for sharing software components among different quantum chemistry packages, which can potentially enable new scientific workflows and/or improve computational efficiency. A software interface was constructed to streamline integration of external two-electron integral computation packages with GAMESS. The ERD solver was interfaced with GAMESS, and improved integral computation times by as much as 28% for a large generally contracted basis set, and 7-15% for more commonly used basis sets. Unfortunately, differences in the integral data formats required by the respective codes requires an additional integral reordering operation which essentially negates any performance benefits. A second integral solver called SIMINT was also integrated with GAMESS. Instead of applying substantial modifications to the GAMESS Hartree-Fock driver to enable integration with SIMINT, a new Hartree-Fock driver was written from scratch (SIMGMS). Compared to the original GAMESS code, serial execution of SIMGMS Fock matrix construction was demonstrated to be faster by 20-26%%. Converged Fock and density matrices computed with SIMGMS can be reordered and

passed to GAMESS for post-Hartree-Fock computations if desired. While the SIMGMS code offers GAMESS the additional flexibility of the SIMINT package, a more minimal interface must be implemented to take advantage of GAMESS features such as the extended Hückel orbital guess, and effective fragment potential electrostatic and polarization terms.

In Chapter 6, the esterification of acetic acid and methanol by heterogeneous catalysis was investigated. Mesoporous silica nanoparticles (MSN) functionalized with propylsulfonic acid were represented with both a minimal all-quantum surface model, and an embedded QM/MM surface model of an extended MSN pore. The adsorption of acetic and methanol was modeled on both isolated single-site catalysts, and pairs of adjacent catalyst sites. It was determined that the minimal models are sufficient for computing optimized structures and adsorption energies at the DFT and MP2 levels of theory for single-catalyst structures. For dual-catalyst structures, the computed adsorption energies and adsorbate orientations largely depend on the relative orientations of the adjacent catalyst sites. Therefore, any proposed reaction mechanisms that require adjacent catalyst sites should be evaluated for a range of relative catalyst orientations. Two single-catalyst stepwise esterification reaction mechanisms were proposed. The mechanisms differ primarily by the mechanism of acetic acid protonation, either directly by methanol, or by the acid catalyst. Both proposed mechanisms are reversible, with similar reaction barrier heights.

In Chapter 7, on-the-fly *ab initio* molecular dynamics simulations with spin-flip time-dependent density functional theory was used to study the photocyclization and photoisomerization mechanisms of *cis*-stilbene following excitation to the $\pi\pi^*$ state. A state

tracking method was developed to distinguish the target electronic states of interest from nearly degenerate states during subsequent dynamics time steps. The dynamics simulations were used to compute a *trans*: dihydrophenanthrene branching ratio of 35:13, which qualitatively reproduces the experimentally measured ratio of 35:10. The favorable ratio of *trans*-stilbene compared to the photocyclization product was attributed to a lower energy barrier corresponding to rotational motion about the central C=C bond from the $\pi\pi^*$ *cis*-stilbene minimum energy structure compared to the kinetic energy of the torsional motion.

Moving forward from the progress presented in this dissertation, future work will be focused on efficiently mapping data management in quantum chemistry software to emerging hardware designs. While there is no consensus as to which hardware architecture will be used to construct the first exascale system, trends such as increasing complex memory hierarchies and manycore massive parallelism are sure to continue. Considering the fascinating science that is already produced within today's computational limitations, the future of computational chemistry is very promising.